STIC
Scientific & Technical Information Center

Search Report
EIC 2100

STIC Database Tracking Number: 238587

To:	From:
To: JENNIFER TO **Location: RND-5B55** **Art Unit: 2195** **Wednesday, September 26, 2007** **Case Serial Number: 10/643769**	**From: ALYSON DILL** **Location: EIC2100** **RND-4B28 / RND-4A18** **Phone: (571)272-3527** **alyson.dill@uspto.gov**

Search Notes

Examiner TO:

A search was executed on the application number listed above. The following database clusters were searched:

- Patent abstract files;
- Patent fulltext files.

. If you require additional information or a refocus of the search contacted:

Alyson Dill
EIC2100
Phone: 571-272-3527
Randolph 4-a-18

Please complete the search feedback form that is attached to the search and return it to EIC2100. Thanks.

Alyson

sira
Search and Information Resources Administration

Table of Contents

With Suggested Relevant References and Page Numbers

Patent Abstract Files

File 347:JAPIO Dec 1976-2006/
File 350:Derwent WPIX 1963-2007

Set	Items	Description
S1	441	(KERNEL? OR PRIVILEGE?) (2W) MODE? ?
S2	8401812	(ARREST? OR BAR OR BARS OR BARRICAD? OR BARRIER? OR B-LOCK??? OR CLOSE? OR HALT OR HALTS OR HALTED OR HINDER? OR HOLD? OR IMPEDE? OR IMPEDIMENT? OR OBSTACLE? OR OBSTRUCTIO-N? OR PREVENT? OR ROADBLOCK? OR STOP???? OR WALL?)
S3	516809	STREAM? OR INSTRUCTION?
S4	362160	(EVENT? ? OR CONTEXT (2N) (SHIFT? OR SWITCH?)OR THREAD? ? OR SYSTEM (2W) CALL???)
S5	54591	S2 (15N) S3
S6	41	S1 (15N) S4
S7	79	S1 (15N) S3
S8	9	S5 AND (S6 OR S7)
S9	5	S8 AND PY=1963:2004
S10	7	S8 AND AY=1963:2004
S11	7	S9 OR S10
S12	7	IDPAT (sorted in duplicate/non-duplicate order)
S13	7	IDPAT (primary/non-duplicate records only)

13/3,K/4 (Item 4 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0013401596 - Drawing available
WPI ACC NO: 2003-491774/200346
XRPX Acc No: N2003-390629

Kernel streaming testing method for use in software filters in multimedia environment, has multiple software filters in virtual circuits receiving multimedia data stream from source module and analysis module
Patent Assignee: MICROSOFT CORP (MICT)
Inventor: CHEN Y; HAGIU C
Patent Family (1 patents, 1 countries)

Patent			Application			
Number	Kind	Date	Number	Kind	Date	Update
US 6526523	B1	20030225	US 1998179647	A	19981027	200346 B

Priority Applications (no., kind, date): US 1998179647 A 19981027

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
US 6526523	B1	EN	11	4	

Alerting Abstract ...DESCRIPTION OF DRAWINGS - The diagram shows a **block** diagram of a computer system **for** testing software filters used in kernel data **streaming**.

Original Publication Data by Authority

Original Abstracts:
...An analysis module receives the processed data stream from the chain of software, filters and produces performance information as a function of the received data **stream**. The source module, **the** analysis module and each software filter operate in a **kernel-mode** and are **configured by** a test tool operating in non-kernel mode. Via the test tool, a user can select appropriate source and analysis modules as well as construct...
...
...

13/3,K/5 (Item 5 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0006369125 - Drawing available
WPI ACC NO: 1993-167876/**199320**
XRPX Acc No: N1993-128473
Fault-tolerant computer system - has all interrupts and traps occurring on primary computer during an epoch buffered by virtual machine monitor
Patent Assignee: DIGITAL EQUIP CORP (DIGI)
Inventor: BALKOVICH E; LAMPSON B; SCHNEIDER F B; THIEL D
Patent Family (2 patents, 18 countries)

Patent		Application

Number Kind Date Number Kind Date Update
WO 1993009494 A1 19930513 WO 1992US9238 A 19921027 199320 B
US 5488716 A 19960130 US 1991783519 A 19911028 199611 E
US 1994182311 A 19940114

Priority Applications (no., kind, date): US 1994182311 A 19940114; US 1991783519 A 19911028

Patent Details
Number Kind Lan Pg Dwg Filing Notes
WO 1993009494 A1 EN 39 7
National Designated States,Original: JP KR
Regional Designated States,Original: AT BE CH DE DK ES FR GB GR IE IT LU MC NL SE
US 5488716 A EN 14 7 Continuation of application US 1991783519

Original Publication Data by Authority

Original Abstracts:
...backup computers. Primary and backup virtual machines running on the computers are controlled by corresponding virtual machine monitors. The virtual machines execute only user-mode **instructions**, while all **kernel-mode instructions** are **trapped and handled** by the virtual machine monitors. Each computer has a recovery register that generates a hardware interrupt each time that a specified number of instructions, called...

...backup computers. Primary and backup virtual machines running on the computers are controlled by corresponding virtual machine monitors. The virtual machines execute only user-mode **instructions**, while all **kernel-mode instructions** are trapped and **handled** by the **virtual machine monitors**. Each computer has a recovery register that generates a hardware interrupt each time that a specified number of instructions, called an epoch, are executed. Prior...
Claims:
...as said interrupts sent by said primary computer, and for delivering said buffered interrupts and traps to said backup computer at predefined points in said **stream** of **instructions** executed **by** said backup computer; backup epoch control means for **stopping** execution **of** said **stream** of **instructions** by **said** backup computer at each of said predefined points until said backup computer receives the control message indicating that said primary computer has completed execution of said stream of instructions through a next one of said predefined points in said **stream** of **instructions** after **the** predefined point at which said backup computer is **stopped**, such **that** said backup computer's execution of said **stream** of **instructions** always **lags** behind the primary computer's

execution of said stream of instructions.Basic Derwent Week: **199320**

13/3,K/6 (Item 6 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0006125044 - Drawing available
WPI ACC NO: 1992-366431/**199244**
XRPX Acc No: N1992-279247
Real-time I-O operation in vector processing computer - using rotating priority interrupt and dedicated real-time interrupt for each processor with access to privileged communication-control modes
Patent Assignee: CRAY RES INC (CRAY)
Inventor: FLUNKER G; LEEDOM G W; PRIBNOW R D; SCHIFFLEGER A J
Patent Family (5 patents, 17 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update
WO 1992017840	A1	19921015	WO 1991US8709	A	19911121	199244 B
EP 577614	A1	19940112	WO 1991US8709	A	19911121	199402 E
			EP 1992904950	A	19911121	
US 5390300	A	19950214	US 1991677647	A	19910328	199512 E
			US 1994200921	A	19940222	
EP 577614	B1	19951004	WO 1991US8709	A	19911121	199544 E
			EP 1992904950	A	19911121	
DE 69113639	E	19951109	DE 69113639	A	19911121	199550 E
			WO 1991US8709	A	19911121	
			EP 1992904950	A	19911121	

Priority Applications (no., kind, date): US 1994200921 A 19940222; US 1991677647 A 19910328

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
WO 1992017840	A1	EN	34	5	
National Designated States,Original: CA JP KR					
Regional Designated States,Original: AT BE CH DE DK ES FR GB GR IT LU NL SE					
EP 577614	A1	EN	2	1	PCT Application WO 1991US8709
					Based on OPI patent WO 1992017840
Regional Designated States,Original: AT BE CH DE DK ES FR GB GR IT LI LU NL SE					
US 5390300	A	EN	14	5	Continuation of application US 1991677647
EP 577614	B1	EN	17	5	PCT Application WO 1991US8709
					Based on OPI patent WO 1992017840
Regional Designated States,Original: AT BE CH DE DK ES FR GB GR IT LI LU NL SE					
DE 69113639	E	DE			PCT Application WO 1991US8709

Application EP 1992904950
Based on OPI patent EP 577614
Based on OPI patent WO 1992017840

Equivalent Alerting Abstract ...designated processing devices is brought, in response to the real-time signal, into a state where the designated processing device is allowed access to a **privileged** operational **mode**, and therefore access to **instructions** corresponding to the **privileged** operational **mode** and to a system privileged communication channel connecting the processing device to the I/O control device...

...signalling device, and signalling the designated processing device from the I/O control device that input data is available using an input signalling device. The **instruction** execution and signalling steps are executed in the **privileged** operational **mode**.

Original Publication Data by Authority

Claims:
...to the real-time command, into a state wherein it is allowed to have access to a priveleged operational mode under control of a predetermined **block** of data being transferred from the shared memory, in order to have access to the **instructions** corresponding to the **privileged** operational **mode** and to the privileged communication channel (42); and (e) bypassing the operating **system** in **that** each designated processor has access to the priveleged communication channels (42) to read and write data thereon and to signal the I/O processor IOS... executing the appropriate real-time instructions; (d) bringing each of the designated processing means, in response to the real-time signal, into a state wherein **the** designated processing means is allowed access to a **privileged** operational **mode**, and therefore access to **instructions** corresponding to the **privileged** operational **mode** and to a system privileged communication **channel** connecting **the** processing means to the **I**/O control means; (**e**) executing **the** real-time **instructions** in the designated processing means; (f) signaling an I/O control means from the designated processing means when output data is available using an output...
...
...

13/3,K/7 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0005457523 - Drawing available

WPI ACC NO: 1991-057867/**199108**
XRPX Acc No: N1991-044805
Central processing unit e.g. for data processing system - has four-level pipeline and interface for coprocessor to provide normal instruction execution
Patent Assignee: NIPPON DIGITAL EQUIP KK (DIGI)
Inventor: LEONARD J S; WILHELM N C
Patent Family (1 patents, 1 countries)

Patent			Application			
Number	Kind	Date	Number	Kind	Date	Update
US 4991078	A	19910205	US 1987101983	A	19870929	199108 B

Priority Applications (no., kind, date): US 1987101983 A 19870929

Alerting Abstract ...The central processing unit has a four stage pipeline, operating with a load/store procedure operation, having a simplified **instruction** on set and providing a user and **kernel mode** of operation. The central processing unit has a data and **instruction** word size of constant (32 bit) width. The addressing technique includes a virtual addressing scheme the virtual addressing scheme is also the mechanism for separation...

Original Publication Data by Authority

Original Abstracts:
...the central processing system, a simplified instruction set and an interface with the coprocessor unit that has a simple and efficient interface with the normal **instruction** execution. The apparatus **implementing** the central processing system is **closely** connected to the **instruction** set. A discussion **of** the implementation of the data processing system is provided. ...
...

Patent Fulltext Files

File 348:EUROPEAN PATENTS 1978-2007
File 349:PCT FULLTEXT 1979-2007

Set	Items	Description
S1	936	(KERNEL? OR PRIVILEGE?) (2W) MODE? ?
S2	2028881	(ARREST? OR BAR OR BARS OR BARRICAD? OR BARRIER? OR B-LOCK??? OR CLOSE? OR HALT OR HALTS OR HALTED OR HINDER? OR HOLD? OR IMPEDE? OR IMPEDIMENT? OR OBSTACLE? OR OBSTRUCTIO-N? OR PREVENT? OR ROADBLOCK? OR STOP???? OR WALL?)
S3	464360	STREAM? OR INSTRUCTION?
S4	652271	(EVENT? ? OR CONTEXT (2N) (SHIFT? OR SWITCH?)OR THREAD? ? OR SYSTEM (2W) CALL???)
S13	45	S5 AND S6 AND S7
S14	38	S13 AND PY=1963:2004
S15	43	S13 AND AY=1963:2004
S16	45	S13 OR S14
S17	45	IDPAT (sorted in duplicate/non-duplicate order)
S18	44	IDPAT (primary/non-duplicate records only)

18/3,K/7 (Item 7 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

01605021
Methods and system for managing computational resources of a coprocessor in a computing system

PATENT ASSIGNEE:
MICROSOFT CORPORATION, (749868), One Microsoft Way, Redmond, WA 98053, (US), (Applicant designated States: all)
INVENTOR:
Wilt, Nicholas P., 1920-205th Place Northeast, Sammamish, Washington 98074, (US)
Nene, Sameer A., 15816 N.E. 46th Ct.,, Redmond, Washington 98052, (US)
Bede, Joseph S., III, 3819 Densmore Avenue N., Seattle, Washington 98103, (US)
LEGAL REPRESENTATIVE:

Grunecker, Kinkeldey, Stockmair & Schwanhausser Anwaltssozietat (100721)
, Maximilianstrasse 58, 80538 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 1326165 A2 030709 (Basic)
APPLICATION (CC, No, Date): EP 2003000153 030103;
PRIORITY (CC, No, Date): US 39036 020104
DESIGNATED STATES: AT; BE; BG; CH; CY; CZ; DE; DK; EE; ES; FI; FR; GB; GR; HU; IE; IT; LI; LU; MC; NL; PT; SE; SI; SK; TR
EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO
INTERNATIONAL PATENT CLASS (V7): G06F-009/38
ABSTRACT WORD COUNT: 92
NOTE:
Figure number on first page: 1A

LANGUAGE (Publication,Procedural,Application): English; English; English

...SPECIFICATION primitive is an object that can be used to synchronize multiple threads' access to shared resources, such as critical sections, mutexes, semaphores or events.

A **thread** is an executable entity that comprises a program counter, a user-mode stack, a **kernel-mode** stack and a set of register values.

A token **stream** is a **stream** of hardware-independent tokens that describe a series of drawing operations. A token stream can be translated by a hardware-specific software component, such as... inefficient and generally does not scale well with multiple instances of a given piece of hardware in a system. As a rule, IN and OUT **instructions** can be executed only in **kernel mode**, the microprocessor mode that allows direct manipulation of hardware. If a user mode **thread** encounters such an instruction, the system generates an exception, which, in the case of a user mode thread attempting to control hardware, usually results in...application activity into DrawPrimitives2 ("DP2") tokens. When the DP2 token stream is submitted, a kernel transition occurs and the driver 314 translates the DP2 token **stream** into hardware-specific commands in **kernel mode**. Fig. 3B does not make any assumptions about whether the driver 314 is in user mode or kernel mode, and in this regard, driver component...non-writeable. Memory writes that overrun the buffer would then cause an exception. User mode code writing into the command buffer could surround the write **instructions** with a structured exception handling **block**, field the exception and return an error to the runtime if the command buffer overflows. The interface to the user mode driver would then be...commands' DDI counterparts into hardware-specific commands and write them into a temporary buffer or buffers (since the exact size of the hardware-specific command **stream** is not known until after the translation has been performed). When the recording is **stopped**, or once the token **stream** has been parsed, the system could then allocate a command buffer or command buffers of suitable size and copy the translated hardware commands into them...could be used. The system of Fig. 7 closely resembles the state of the art in DIRECT3D(R)

driver structure, in that a DrawPrimitives2 token **stream** is passed to the **kernel mode** driver 705. The main difference is that the system of Fig. 7 contemplates OS-arbitrated scheduling of the command buffers, while DIRECT3D(R) currently does...

...CLAIMS including preempting by the at least one coprocessor upon the occurrence of an external event.

19. A method according to claim 18, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the at least one coprocessor.

20. A method according to claim 18, wherein the host processor is interrupted to coordinate scheduling...at least one coprocessor upon the occurrence of an external event.

45. At least one computer readable medium according to claim 44, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the at least one coprocessor.

46. At least one computer readable medium according to claim 27, wherein the host processor is...preempted by the at least one coprocessor upon the occurrence of an external event.

70. A computing device according to claim 69, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the at least one coprocessor.

71. A computing device according to claim 52, wherein the host processor is interrupted to coordinate...

18/3,K/8 (Item 8 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

01059137
METHOD AND APPARATUS FOR SELECTING THREAD SWITCH EVENTS IN A MULTITHREADED PROCESSOR

PATENT ASSIGNEE:
International Business Machines Corporation, (200128), New Orchard Road, Armonk, NY 10504, (US), (Proprietor designated states: all)

INVENTOR:
BORKENHAGEN, John, Michael, 1359 Westhill Drive S.W., Rochester, MN 55902 , (US)
EICKEMEYER, Richard, James, 5277 Howard Street N.W., Rochester, MN 55901, (US)
FLYNN, William, Thomas, 2516 14th Avenue S.W., Rochester, MN 55902, (US)
LEVENSTEIN, Sheldon, Bernard, 1608 7th Street N.E., Rochester, MN 55906, (US)
WOTTRENG, Andrew, Henry, 4224 Manor View Drive N.W., Rochester, MN 55901,

(US)
LEGAL REPRESENTATIVE:
Burt, Roger James, Dr. et al (52152), IBM United Kingdom Limited
Intellectual Property Department Hursley Park, Winchester Hampshire
SO21 2JN, (GB)
PATENT (CC, No, Kind, Date): EP 1029269 A1 000823 (Basic)
EP 1029269 B1 030924
WO 99021081 990429
APPLICATION (CC, No, Date): EP 98953515 981014; WO 98US21716 981014
PRIORITY (CC, No, Date): US 958716 971023
DESIGNATED STATES: GB; IE
INTERNATIONAL PATENT CLASS (V7): G06F-009/38; G06F-009/46
NOTE:
No A-document published by EPO
LANGUAGE (Publication,Procedural,Application): English; English; English
...SPECIFICATION processing system according to the present invention.
Figure 3 illustrates a block diagram of the storage control unit of Figure 2.
Figure 4 illustrates a **block** diagram of the thread switch logic, the storage control unit and the **instruction** unit of Figure 2.
Figure 5 illustrate the changes of state of a thread as the thread experiences different thread switch events shown in Figure...which allows up to a programmable maximum number of thread switches called the forward progress threshold value. After that maximum number of thread switches, an **instruction** must be completed before switching can occur again. In this way, thrashing is **prevented**. Forward progress count register 420 may actually be bits 30:31 in the thread switch control register 410 or a software programmable forward progress threshold...

...610, bits 15:17 in thread state register 442 pertaining to thread T0 are reset to state 111. Execution of this thread is attempted in **block** 620 and the state changes to 000. If an **instruction** successfully executes on thread T0, the state of thread T0 returns to 111 and remains so. If, however, thread T0 cannot execute an instruction, a...itself to change the priority of itself:
tsop 1 or 1,1,1 - Switch to dormant thread
tsop 2 or 1,1,1 - Set active **thread** to LOW priority

- Switch to dormant **thread**

- NOTE: Only valid in **privileged mode** unless TSC(19)=1
tsop 3 or 2,2,2 - Set active **thread** to MEDIUM priority
tsop 4 or 3,3,3 - Set active **thread** to HIGH priority

- NOTE: Only valid in **privileged mode**
Instructions tsop 1 and tsop 2 can be the same **instruction** as embodied herein as or 1,1,1 but they can also be separate instructions. These instructions interact with bits 19 and 21 of the...

...thread switch control register 320 is zero, the priority for both threads is set to medium and the effect of the or x,x,x **instructions** on the priority is **blocked**. If an external interrupt request is active, and if the corresponding thread's priority is low, that thread's priority is set to medium.
The...

18/3,K/9 (Item 9 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00959156
Method and computer program product for interconnecting software drivers in kernel mode

PATENT ASSIGNEE:
MICROSOFT CORPORATION, (749861), One Microsoft Way, Redmond, Washington 98052-6399, (US), (applicant designated states: AT;BE;CH;DE;DK;ES;FI;FR;GB;GR;IE;IT;LI;LU;MC;NL;PT;SE)
INVENTOR:
Shaw, George H. J., 21213 NE 186 th Street, Woodinville WA 98072, (US)
O'Rourke, Thomas, Riekonmarjatie 29, 90800 Oulu, (FI)
Woodruff, Bryan A., 1020 SW 10th Street, New Bend WA 98045, (US)
LEGAL REPRESENTATIVE:
Belcher, Simon James et al (58311), Urquhart-Dykes & Lord Tower House Merrion Way, Leeds LS2 8PA, (GB)
PATENT (CC, No, Kind, Date): EP 871114 A2 981014 (Basic)
EP 871114 A3 981216
APPLICATION (CC, No, Date): EP 97304296 970619;
PRIORITY (CC, No, Date): US 825856 970404
DESIGNATED STATES: DE; FR; GB
INTERNATIONAL PATENT CLASS (V7): G06F-009/46; G06F-013/10;
ABSTRACT WORD COUNT: 153

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS A (English) 9842 895
SPEC A (English) 9842 11279
Total word count - document A 12174
Total word count - document B 0
Total word count - documents A + B 12174

...SPECIFICATION of easily using kernel mode drivers, used throughout this application, comprises graph building functionality that allows a user to select and connect together different processing **blocks**, called filters, to successively manipulate a **stream** of multimedia data.
The data typically is a series of samples representing sound or video,

and the processing blocks may include decompression processing for compressed...
...so that the graph builder portion of the program may interconnect and control their operation and be responsive to user input and rearrangement of processing **blocks**. Because of the consistent **stream** nature of multimedia data and the generation of large quantities of data, performance is a critical issue. In a general purpose operating system, system performance...44 will query the drivers known in order to then make interconnections according to data format and connection format to effectuate the rendering entirely in **kernel mode**.
Furthermore, the controlling agent will receive notifications of important **events** so that it may exercise control as necessary. Examples of such events would include end of processing, a data starvation situation, etc.

In this configuration...discussion purposes hereinafter transitioning to the acquire state will accomplish the same purpose with respect to stack depth parameter adjustment as transitioning out of the **stop** state.

Once all pin instances are in the acquire state, **stream** reads and writes may be issued to the filter graph. It is interesting to note that the system explained herein allows connection of associated input... output pin instance 208 to finalize the connection.

At step 197, the third party controlling agent 170 will transition each connection pin instance from the **stop** state to the acquire state in preparation for **streamed** data processing through the filter graph. To correctly set the stack depth parameter in each of the device objects for the respective filters, it is...

...CLAIMS a plurality of kernel mode data processing components including an originating component and a terminating component, the originating component reading data samples of a data **stream** from the data source; and

kernel mode component interconnections between the data processing components to route the data samples from the originating component to the terminating component.

11. A data processing system...

18/3,K/10 (Item 10 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00957724
Method and computer program product for synchronizing the processing of multiple data streams and matching disparate processing rates using a standardized clock mechanism

PATENT ASSIGNEE:
MICROSOFT CORPORATION, (749861), One Microsoft Way, Redmond, Washington

98052-6399, (US), (Applicant designated States: all)
INVENTOR:
Shaw, George H.J., 21213 NE 186th Street, Woodinville, WA 98072, (US)
O'Rourke, Thomas J., Riekonmarjatie 29, 90800 Oulu, (FI)
Woodruff, Bryan A., 1020 SW 10th Street, New Bend, WA 98045, (US)
LEGAL REPRESENTATIVE:
Alton, Andrew et al (97091), Urquhart-Dykes & Lord Tower House Merrion Way, Leeds LS2 8PA, (GB)
PATENT (CC, No, Kind, Date): EP 869443 A2 981007 (Basic)
EP 869443 A3 010509
APPLICATION (CC, No, Date): EP 97304294 970619;
PRIORITY (CC, No, Date): US 826560 970404
DESIGNATED STATES: DE; FR; GB
EXTENDED DESIGNATED STATES: AL; LT; LV; RO; SI
INTERNATIONAL PATENT CLASS (V7): G06F-017/30
ABSTRACT WORD COUNT: 227
NOTE:
Figure number on first page: 2

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	9841	1278
SPEC A	(English)	9841	21034

Total word count - document A 22312
Total word count - document B 0
Total word count - documents A + B 22312

...SPECIFICATION stream.
Another method of synchronization utilizes a clock mechanism that provides a time value that is based on presentation timestamp values on a "master" data **stream** while the **stream** is being processed. When the master **stream** is paused or otherwise **stopped**, the time value alternates to be based off a physical or hardware clock such as the PC clock. While the processor of a second data...

...on the physical clock.
Since event notification is based on the physical clock, such events may continue to occur even when processing is paused or **stopped** on the master data or media **stream**. Notifications may be temporarily disabled but at the cost of extra processing overhead and increased processing code complexity. Even when notifications are specifically disabled, timing easily using kernel mode drivers, used throughout this application, comprises graph building functionality that allows a user to select and connect together different processing **blocks**, called filters, to successively manipulate a **stream** of multimedia data. The data typically is a series of samples representing sound or video and the processing blocks may include decompression processing for compressed ...

...so that the graph builder portion of the program may interconnect and control their operation and be responsive to user input and rearrangement of processing **blocks**. Because of the consistent **stream** nature of multimedia data and the generation of large quantities of data, performance is a critical issue. In a general purpose operating system, system performance...

...clock mechanism is provided. The clock mechanism can be implemented in a variety of different ways and is explained throughout in the context of interconnectable **kernel mode** drivers providing efficient data **stream** processing.

The clock mechanism of the present invention maintains three different time values and makes them available to other components. The time values may be...

...can be used by other processing components processing other data streams for synchronization. Since the positional time value is based on processing of a data **stream**, when the processing of the underlying data **stream stops** the clock does as well. Furthermore, timing event notifications will also be frozen on an absolute time basis, but will continue to be perfectly timed...showing the operation of buffer allocators and the actual data transferring between buffers for the system shown in Figure 12.

Figure 14 is a logical **block** diagram illustrating how two live data **streams** can be synchronized to a single master clock mechanism.

Figures 15A and 15B are logical **block** diagrams showing the live audio system of Figure 14 as implemented using the interconnected filter system explained in detail in connection with Figure 12. Figure...44 will query the drivers known in order to then make interconnections according to data format and connection format to effectuate the rendering entirely in **kernel mode**. Furthermore, the controlling agent will receive notifications of important **events** so that it may exercise control as necessary. Examples of such events would include end of processing, a data starvation situation, etc.

In this configuration...discussion purposes hereinafter transitioning to the acquire state will accomplish the same purpose with respect to stack depth parameter adjustment as transitioning out of the **stop** state.

Once all pin instances are in the acquire state, **stream** reads and writes may be issued to the filter graph. It is interesting to note that the system explained herein allows connection of associated input... output pin instance 208 to finalize the connection.

At step 197, the third party controlling agent 170 will transition each connection pin instance from the **stop** state to the acquire state in preparation for **streamed** data processing through the filter graph. To correctly set the stack depth parameter in each of the device objects for the respective filters, it is...call (which may be done during a

DPC), the allocator returns a pointer to the frame if one is available or returns null immediately. The **kernel mode** requester may then wait for a free **event** notification to know that there is a free frame available. Upon receipt of this notification, the kernel mode requester will attempt the allocate request again...synchronized.

In order to accomplish this, one data stream is chosen to be the master. For purposes of this disclosure and explanation, the live audio **stream** 298 has been arbitrarily chosen to be the master data **stream**. Note that nothing **prevents** the live video **stream** 308 from being chosen as the master or any other **stream** of data for that matter. Furthermore, an external clock, based on a data stream entirely external to the filter graph, may be used to synchronize based on the time interval information in the live audio **stream** 298, no overhead is necessary to manage **stopped** or paused states since media time advances ("ticks") only when the live audio **stream** 298 is being processed at the audio renderer 296.

It should be noted that the video stream 308 may also need to be rate matched...instances and output pin instances in order to make the interconnections between the respective filters and it is the controlling agent that makes the entire **kernel mode** filter graph topology.

The live audio **stream** 352 is initially brought into the filter graph topology by the audio reader filter 354 as represented by arrow 356. The data is decompressed by...architecture while allowing unbounded ability for the filter developer to customize and add additional capabilities.

While a property set allows other processing components, such as **kernel mode** filters, to query relevant time properties, clock mechanisms may also support an **event** set so that notifications may be sent directly to interested clock mechanism clients by way of event notification IRPs. The handle of a file object...

...clock mechanism will know where to send event notifications. Optionally, direct function call handles may be used so that DPC level processing can occur between **kernel mode** entities for achieving higher performance.

Table 5 below, a number of possible notification **events** make up an event set that can be supported by compliant drivers. A third party controlling agent may interconnect or register those entities for receiving...

...336 of the video renderer filter 332 as a slave processing component.

Note also, that the media time value used depends upon the master data **stream** processing. Should media rendering be paused or **stopped**, then time would effectively **stop** for synchronization purposes.

The slave processing component or filter will compare the master clock media time value to the slave data stream media time value...the local host time rather than the PC time.

Applying the method for translation illustrated in the flow chart of Figure 16C to the interconnected **kernel mode** filters for processing and rendering a live audio and video **stream** as shown in Figures 15A and 15B, it would be the input pin instance 336 on the video renderer filter 332 that would receive the...

...CLAIMS A method as claimed in claim 17, in which the designated time value is a physical time value based upon a hardware oscillator.

20. A **kernel mode** data processing system allowing synchronous processing of two data **streams** comprising:

a first data source;

a first plurality of **kernel mode** data processing components including an originating component and a terminating component, the originating component reading data samples of a first data stream originating from the...

18/3,K/11 (Item 11 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00952420
A cache coherency mechanism
Cachespeicherkoharenzmechanismus
Mecanisme de coherence d'antememoire
PATENT ASSIGNEE:
STMicroelectronics Limited, (1828183), 1000 Aztec West, Almondsbury, Bristol BS32 4SQ, (GB), (applicant designated states: DE;FR;GB;IT)
INVENTOR:
Barnaby, Catherine, 18 Roundways, Coalpit Heath, Bristol BS17 2QY, (GB)
Fel, Bruno, 14 rue du Moucherotte, 38360 Sassenage, (FR)
Farrall, Glen, 157 Long Ashton Road, Long Ashton, Bristol BS41 9JQ, (GB)
LEGAL REPRESENTATIVE:
Driver, Virginia Rozanne et al (58902), Page White & Farrer 54 Doughty Street, London WC1N 2LS, (GB)
PATENT (CC, No, Kind, Date): EP 863464 A1 980909 (Basic)
APPLICATION (CC, No, Date): EP 98301617 980304;
PRIORITY (CC, No, Date): GB 9704542 970305
DESIGNATED STATES: DE; FR; GB; IT
INTERNATIONAL PATENT CLASS (V7): G06F-012/08
ABSTRACT WORD COUNT: 167
LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	9837	845
SPEC A	(English)	9837	5615

Total word count - document A 6460
Total word count - document B 0
Total word count - documents A + B 6460

...SPECIFICATION mapped. However, other associativities are possible.
The invention also provides a computer system comprising:
a processor for running a process by executing a sequence of **instructions**;
a main memory which **holds** said **instructions** and data for said **instructions**; and
a cache connected in a memory access path between the processor and the main memory and having a plurality of storage locations, wherein a...

...in main memory are held at that location in the cache.
In the preferred embodiment, the processor has a user mode of operation and a **privileged (kernel) mode** of operation. Cache coherency **instructions** are executable in the user mode.
For a better understanding of the present invention and to show how the same may be carried into effect...have their partition indicator bits set for that partition. However, a user may alter partitions by requesting that the cache partitions be altered. In that **event**, the CPU 2 would change to **kernel mode** to implement the request, change the TLB entries accordingly and then return to the user mode to allow the user to continue. Thus, a user...single cache access circuit which performs both look-up and refill.
As can be seen in Figures 2 and 8, each location in the cache **holds** an address in main memory and the item (data or **instruction**) from that address in main memory. It is not necessary for the whole of the memory address to be held at the cache location. For ...

...CLAIMS or each cache partition, when present, is direct mapped.
10. A computer system comprising:
a processor for running a process by executing a sequence of **instructions**;
a main memory which **holds** said **instructions** and data for said **instructions**; and
a cache connected in a memory access path between the processor and the main memory and having a plurality of storage locations, wherein a...

...at that location in the cache.
11. A computer system according to claim 10, wherein the processor has a user mode of operation and a **privileged mode** of operation and wherein the cache coherency **instruction** is executed in the user mode.
12. A method of modifying the coherency status of the contents of a cache with respect to items held...

18/3,K/12 (Item 12 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00723208
Fault-tolerant computer system with online reintegration and shutdown/restart
Fehlertolerantes Rechnersystem mit Online-Wiedereinfugung und Abschaltung/Start
Systeme de calculateur tolerant les fautes avec reintegration et arret d'ingence/reprise en ligne
PATENT ASSIGNEE:
TANDEM COMPUTERS INCORPORATED, (524031), 10435 N. Tantau Avenue, Cupertino, California 95014-0709, (US), (applicant designated states: AT;BE;CH;DE;DK;ES;FR;GB;GR;IT;LI;LU;NL;SE)
INVENTOR:
Debacker, Kenneth C., 15702 Monona Cove, Austin, Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin, Texas 78752, (US)
Webster, Phil, 9743 Anderson Village Dr., Austin, Texas 78758, (US)
Aldridge, Dave, 20542 Highland Lake Drive, Lage Vista, Texas 78645, (US)
Norwood, Peter C., 2200 Klattenhoff Dr., Austin, Texas 78728, (US)
Jewett, Douglas E., 12401 Wycliff Lane, Austin, Texas 78727, (US)
Bereiter, Tom, 4613 Shoalwood, Austin, Texas 78756, (US)
Vetter, Brian, 8501 Potosi Cove, Austin, Texas 78717, (US)
Banton, Randall G., 1805 Stoneridge Road, Austin, Texas 78746, (US)
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown, Texas 78626, (US)
Westbrook, Donald C., ., (deceased), (US)
LEGAL REPRESENTATIVE:
Liesegang, Roland, Dr.-Ing. (7741), FORRESTER & BOEHMERT Franz-Joseph-Strasse 38, 80801 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 683456 A1 951122 (Basic)
EP 683456 B1 980722
APPLICATION (CC, No, Date): EP 95111528 901218;
PRIORITY (CC, No, Date): US 455065 891222; US 461250 900105; US 461402 900105
DESIGNATED STATES: AT; BE; CH; DE; DK; ES; FR; GB; GR; IT; LI; LU; NL; SE
RELATED PARENT NUMBER(S) - PN (AN):
EP 433979 (EP 901245829)
INTERNATIONAL PATENT CLASS (V7): G06F-011/14;
ABSTRACT WORD COUNT: 139

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS B (English) 9830 1353
CLAIMS B (German) 9830 1337
CLAIMS B (French) 9830 1443
SPEC B (English) 9830 32820
Total word count - document A 0
Total word count - document B 36953
Total word count - documents A + B 36953

...SPECIFICATION local bus are connected to pipeline and bus control circuitry 87, driven from the internal bus structure 81 and the control logic 82.

The microprocessor **block** 75 in the processor 40 is of the RISC type in that most **instructions** execute in one machine cycle, and the instruction set uses register-to-register and load/store instructions rather than having complex instructions involving memory references... more difficult, and would degrade performance; also, use of the write buffer 50 serves to decouple the processors, and would be much less effective with **close** coupling of the processors. Likewise, the high-performance resulting from using **instruction** and data caches, and virtual memory management with the TLBs 83, would be more difficult to implement if close coupling were used, and performance would... provided in many machines. The processor 40 is configured to operate normally in the user mode until an exception is detected forcing it into the **kernel mode**, where it remains until a restore from exception (RFE) **instruction** is executed. The manner in which the memory addresses are translated or mapped depends upon the operating mode of the microprocessor, which is defined by...

...The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to **hold** user code and data, but the operating system often needs to reference this same code or data. The kseg0 area 152 is a 512-MByte...

...This kseg2 area differs from the kuseg area 150 only in that it is not accessible in the user mode, but instead only in the **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for somedynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather than...

18/3,K/13 (Item 13 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00719992
Fault-tolerant computer system with I/O function capabilities.
Fehlertolerantes Computersystem mit Ein-/Ausgabefunktionsfahigkeiten.
Systeme informatique insensible aux defaillances avec possibilites de fonction entree/sortie.
PATENT ASSIGNEE:
TANDEM COMPUTERS INCORPORATED, (524033), 10435 N. Tantau Avenue, Cupertino, California 95014, (US), (applicant designated states: AT;DE;FR;GB;IT;SE)

INVENTOR:
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown, Texas 78626, (US)
Banton, Randall G., 1805 Stoneridge Road, Austin, Texas 78746, (US)
Jewett, Douglas E., 12401 Wycliff Lane, Austin, Texas 78727, (US)
Norwood, Peter C., 2200 Klattenhoff Dr., Austin, Texas 78728, (US)
Debacker, Kenneth C., 15702 Monona Cove, Austin, Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin, Texas 78758, (US)
Allison, John David, 1406 Windsor Road, No. 202, Austin, Texas 78703, (US)
Horst, Robert W., 2804 Robeson Park Dr., Champaige, Illinois 61821, (US)
LEGAL REPRESENTATIVE:
Altenburg, Udo, Dipl.-Phys. et al (1269), Patent- und Rechtsanwalte Bardehle . Pagenberg . Dost . Altenburg . Geissler . Isenbruck Postfach 86 06 20, 81633 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 681239 A2 951108 (Basic)
EP 681239 A3 960124
APPLICATION (CC, No, Date): EP 95111006 891208;
PRIORITY (CC, No, Date): US 282629 881209; US 283574 881213
DESIGNATED STATES: AT; DE; FR; GB; IT; SE
RELATED PARENT NUMBER(S) - PN (AN):
EP 372579 (EP 891227084)
INTERNATIONAL PATENT CLASS (V7): G06F-011/16; G06F-011/18;
ABSTRACT WORD COUNT: 232

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS A (English) EPAB95 695
SPEC A (English) EPAB95 20127
Total word count - document A 20822
Total word count - document B 0
Total word count - documents A + B 20822

...SPECIFICATION local bus are connected to pipeline and bus control circuitry 87, driven from the internal bus structure 81 and the control logic 82.

The microprocessor **block** 75 in the processor 40 is of the RISC type in that most **instructions** execute in one machine cycle, and the instruction set uses register-to-register and load/store instructions rather than having complex instructions involving memory references... more difficult, and would degrade performance; also, use of the write buffer 50 serves to decouple the processors, and would be much less effective with **close** coupling of the processors. Likewise, the high-performance resulting from using **instruction** and data caches, and virtual memory management with the TLBs 83, would be more difficult to implement if close coupling were used, and performance would...then presents the interrupt to the processor chip 40. The result is that all of the processor chips received the external interrupt request at identical **instructions**, and the information saved in the

holding registers is not needed.
Holding Register:
In the interrupt scenario presented above with reference to Figure 15, the voter 136 uses a holding register 138 to save some state information ...provided in many machines. The processor 40 is configured to operate normally in the user mode until an exception is detected forcing it into the **kernel mode**, where it remains until a restore from exception (RFE) **instruction** is executed. The manner in which the memory addresses are translated or mapped depends upon the operating mode of the microprocessor, which is defined by...

...The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to **hold** user code and data, but the operating system often needs to reference this same ...This kseg2 area differs from the kuseg area 150 only in that it is not accessible in the user mode, but instead only in the **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for some dynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather...

18/3,K/14 (Item 14 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00426745
Fault-tolerant computer system with/config filesystem.
Fehlertolerantes Rechnersystem mit Online-Wiedereinfugung und Abschaltung/Start.
Systeme de calculateur tolerant les fautes avec reintegration et arret-d'urgence/reprise en ligne.
PATENT ASSIGNEE:
TANDEM COMPUTERS INCORPORATED, (524031), 10435 N. Tantau Avenue, Cupertino, California 95014-0709, (US), (applicant designated states: AT;BE;CH;DE;DK;ES;FR;GB;GR;IT;LI;LU;NL;SE)
INVENTOR:
Jewett, Douglas E., 12401 Wycliff Lane, Austin, Texas 78727, (US)
Bereiter, Tom, 4613 Shoalwood, Austin, Texas 78756, (US)
Vetter, Brian, 8501 Potosi Cove, Austin, Texas 78717, (US)
Banton, Randall G., 1805 Stoneridge Road, Austin, Texas 78746, (US)
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown, Texas 78626, (US)
Westbrook, Donald C., , deceased, (US)
Fey, Kyran W., Jr., 507 Hickory Ridge Trail, Pfluggerville, Texas 78660, (US)
Pozdro, John, 4617 Sinclair Avenue, Austin, Texas 78756, (US)

Debacker, Kenneth C., 15702 Monona Cove, Austin, Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin, Texas 78752, (US)
Webster, Phil, 9743 Anderson Village Dr., Austin, Texas 78758, (US)
Aldridge, Dave, 20542 Highland Lake Drive, Lage Vista, Texas 78645, (US)
Norwood, Peter C., 2200 Klattenhoff Dr., Austin, Texas 78728, (US)
LEGAL REPRESENTATIVE:
Liesegang, Roland, Dr.-Ing. et al (7741), FORRESTER & BOEHMERT
Franz-Joseph-Strasse 38, W-8000 Munchen 40, (DE)
PATENT (CC, No, Kind, Date): EP 433979 A2 910626 (Basic)
EP 433979 A3 930526
APPLICATION (CC, No, Date): EP 90124582 901218;
PRIORITY (CC, No, Date): US 455065 891222; US 455127 891222; US 455218 891222; US 461250 900105; US 461402 900105
DESIGNATED STATES: AT; BE; CH; DE; DK; ES; FR; GB; GR; IT; LI; LU; NL; SE
INTERNATIONAL PATENT CLASS (V7): G06F-011/16; G06F-011/14;
ABSTRACT WORD COUNT: 356

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	EPABF1	4317
SPEC A	(English)	EPABF1	29454
Total word count - document A			33771
Total word count - document B			0
Total word count - documents A + B			33771

...SPECIFICATION local bus are connected to pipeline and bus control circuitry 87, driven from the internal bus structure 81 and the control logic 82.

The microprocessor **block** 75 in the processor 40 is of the RISC type in that most **instructions** execute in one machine cycle, and the instruction set uses register-to-register and load/store instructions rather than having complex instructions involving memory references... more difficult, and would degrade performance; also, use of the write buffer 50 serves to decouple the processors, and would be much less effective with **close** coupling of the processors. Likewise, the high-performance resulting from using **instruction** and data caches, and virtual memory management with the TLBs 83, would be more difficult to implement if close coupling were used, and performance would... provided in many machines. The processor 40 is configured to operate normally in the user mode until an exception is detected forcing it into the **kernel** **mode**, where it remains until a restore from exception (RFE) **instruction** is executed. The manner in which the memory addresses are translated or mapped depends upon the operating mode of the microprocessor, which is defined by...The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to

hold user code and data, but the operating system often needs to reference this same code or data. The kseg0 area 152 is a 512-MByte...

...This kseg2 area differs from the kuseg area 150 only in that it is not accessible in the user mode, but instead only in the **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for some dynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather...

18/3,K/15 (Item 15 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00355848
Synchronization of fault-tolerant computer system having multiple processors.
Synchronisation eines fehlertoleranten Rechnersystems mit mehreren Prozessoren.
Synchronisation de systeme de calculateur a multiples processeurs a tolerance de fautes.
PATENT ASSIGNEE:
TANDEM COMPUTERS INCORPORATED, (524031), 10435 N. Tantau Avenue, Cupertino California 95014, (US), (applicant designated states: AT;DE;FR;GB;IT;SE)
INVENTOR:
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown Texas 78626, (US)
Jewett, Douglas E., 12401 Wycliff Lane, Austin Texas 78727, (US)
Southworth, Richard A., 10008 Brandywine Circle, Austin Texas 78750, (US)
Debacker, Kenneth C., 15702 Monona Cove, Austin Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin Texas 78758, (US)
Allison, John David, 1406 Windsor Road, No. 202, Austin Texas 78703, (US)
Horst, Robert W., 2804 Robeson Park Drive, Champaign Illinois 61821, (US)
LEGAL REPRESENTATIVE:
Altenburg, Udo, Dipl.-Phys. et al (1268), Patent- und Rechtsanwalte Bardehle-Pagenberg-Dost-Altenburg Frohwitter-Geissler & Partner Postfach 86 06 20, D-8000 Munchen 86, (DE)
PATENT (CC, No, Kind, Date): EP 372580 A2 900613 (Basic)
EP 372580 A3 910724
APPLICATION (CC, No, Date): EP 89122709 891208;
PRIORITY (CC, No, Date): US 282538 881209; US 283141 881209; US 283139 881209
DESIGNATED STATES: AT; DE; FR; GB; IT; SE
INTERNATIONAL PATENT CLASS (V7): G06F-011/16; G06F-011/18;
ABSTRACT WORD COUNT: 157

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:

Available Text Language Update Word Count
CLAIMS A (English) EPABF1 2357
SPEC A (English) EPABF1 24091
Total word count - document A 26448
Total word count - document B 0
Total word count - documents A + B 26448

...SPECIFICATION local bus are connected to pipeline and bus control circuitry 87, driven from the internal bus structure 81 and the control logic 82.

The microprocessor **block** 75 in the processor 40 is of the RISC type in that most **instructions** execute in one machine cycle, and the instruction set uses register-to-register and load/store instructions rather than having complex instructions involving memory ...more difficult, and would degrade performance; also, use of the write buffer 50 serves to decouple the processors, and would be much less effective with **close** coupling of the processors. Likewise, the high-performance resulting from using **instruction** and data caches, and virtual memory management with the TLBs 83, would be more difficult to implement if close coupling were used, and performance would...then presents the interrupt to the processor chip 40. The result is that all of the processor chips received the external interrupt request at identical **instructions**, and the information saved in the **holding** registers is not needed.

Holding Register:

In the interrupt scenario presented above with reference to Figure 15, the voter 136 uses a holding register 138 to save some state information ...provided in many machines. The processor 40 is configured to operate normally in the user mode until an exception is detected forcing it into the **kernel mode**, where it remains until a restore from exception (RFE) **instruction** is executed. The manner in which the memory addresses are translated or mapped depends upon the operating mode of the microprocessor, which is defined by...

...The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to **hold** user code and data, but the operating system often needs to reference this same code or data. The kseg0 area 152 is a 512-MByte...This kseg2 area differs from the kuseg area 150 only in that it is not accessible in the user mode, but instead only in the **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for some dynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather...

18/3,K/16 (Item 16 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00355847
High-performance computer system with fault-tolerant capability; method for operating such a system
Hochleistungsrechnersystem mit fehlertoleranter Fahigkeit; Verfahren zum Betrieb desselben
Systeme de calculateur a haute performance a capacite de tolerance de fautes; methode pour operer un tel systeme
PATENT ASSIGNEE:
TANDEM COMPUTERS INCORPORATED, (524031), 10435 N. Tantau Avenue, Cupertino, California 95014-0709, (US), (applicant designated states: AT;DE;FR;GB;IT;SE)
INVENTOR:
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown Texas 78626, (US)
Banton, Randall G., 1805 Stoneridge Road, Austin Texas 78746, (US)
Jewett, Douglas E., 12401 Wycliff Lane, Austin Texas 78727, (US)
Norwood, Peter C., 2200 Klattenhoff Drive, Austin Texas 78728, (US)
Debacker, Kenneth C., 15702 Monona Cove, Austin Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin Texas 78758, (US)
Allison, John David, 1406 Windsor Road, Nr. 202, Austin Texas 78703, (US)
Horst, Robert W., 2804 Robeson Park Drive, Champaign Illinois 61821, (US)
LEGAL REPRESENTATIVE:
Altenburg, Udo, Dipl.-Phys. et al (1268), Patent- und Rechtsanwalte, Bardehle . Pagenberg . Dost . Altenburg . Frohwitter . Geissler & Partner, Galileiplatz 1, 81679 Munchen, (DE)
PATENT (CC, No, Kind, Date): EP 372579 A2 900613 (Basic)
EP 372579 A3 910724
EP 372579 B1 971001
APPLICATION (CC, No, Date): EP 89122708 891208;
PRIORITY (CC, No, Date): US 282629 881209; US 283574 881213
DESIGNATED STATES: AT; DE; FR; GB; IT; SE
INTERNATIONAL PATENT CLASS (V7): G06F-011/16; G06F-011/18;
ABSTRACT WORD COUNT: 199

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS B (English) 9709W4 920
CLAIMS B (German) 9709W4 851
CLAIMS B (French) 9709W4 991
SPEC B (English) 9709W4 20185
Total word count - document A 0
Total word count - document B 22947
Total word count - documents A + B 22947
...SPECIFICATION local bus are connected to pipeline and bus control circuitry 87, driven from the internal bus structure 81 and the control

logic 82.

The microprocessor **block** 75 in the processor 40 is of the RISC type in that most **instructions** execute in one machine cycle, and the instruction set uses register-to-register and load/store instructions rather than having complex instructions involving memory references... more difficult, and would degrade performance; also, use of the write buffer 50 serves to decouple the processors, and would be much less effective with **close** coupling of the processors. Likewise, the high-performance resulting from using **instruction** and data caches, and virtual memory management with the TLBs 83, would be more difficult to implement if close coupling were used, and performance would...then presents the interrupt to the processor chip 40. The result is that all of the processor chips received the external interrupt request at identical **instructions**, and the information saved in the **holding** registers is not needed.

Holding Register:

In the interrupt scenario presented above with reference to Figure 15, the voter 136 uses a holding register 138 to save some state information ...provided in many machines. The processor 40 is configured to operate normally in the user mode until an exception is detected forcing it into the **kernel mode**, where it remains until a restore from exception (RFE) **instruction** is executed. The manner in which the memory addresses are translated or mapped depends upon the operating mode of the microprocessor, which is defined by...

...The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to **hold** user code and data, but the operating system often needs to reference this same code or data. The kseg0 area 152 is a 512-MByte **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for some dynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather...

18/3,K/17 (Item 17 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00355846
Memory management in high-performance fault-tolerant computer system.
Speicherverwaltung in fehlertolerantem Hochleistungsrechnersystem.
Gestion de memoire dans un systeme a calculateur a haute performance a tolerance de fautes.
PATENT ASSIGNEE:

TANDEM COMPUTERS INCORPORATED, (524031), 10435 N. Tantau Avenue, Cupertino California 95014, (US), (applicant designated states: AT;DE;FR;GB;IT;SE)
INVENTOR:
Cutts, Richard W., Jr., 1312 Elm Street, Georgetown Texas 78626, (US)
Peet, Charles E., Jr., 11920 Oakbrook Drive, Austin Texas 78753, (US)
Jewett, Douglas E., 12401 Wycliff Lane, Austin Texas 78727, (US)
Debacker, Kenneth C., 15702 Monona Cove, Austin Texas 78717, (US)
Mehta, Nikhil A., 1715 Prairie Hen Cove, Austin Texas 78758, (US)
Allison, John David, 1406 Windsor Road, Nr. 202, Austin Texas 78703, (US)
Horst, Robert W., 2804 Robeson Park Drive, Champaige Illinois 61821, (US)
LEGAL REPRESENTATIVE:
Altenburg, Udo, Dipl.-Phys. et al (1268), Patent- und Rechtsanwalte Bardehle-Pagenberg-Dost-Altenburg Frohwitter-Geissler & Partner Postfach 86 06 20, D-8000 Munchen 86, (DE)
PATENT (CC, No, Kind, Date): EP 372578 A2 900613 (Basic)
EP 372578 A3 920115
APPLICATION (CC, No, Date): EP 89122707 891208;
PRIORITY (CC, No, Date): US 282469 881209; US 282540 881209; US 283573 881213
DESIGNATED STATES: AT; DE; FR; GB; IT; SE
INTERNATIONAL PATENT CLASS (V7): G06F-011/16; G06F-011/18;
ABSTRACT WORD COUNT: 301

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS A (English) EPABF1 1931
SPEC A (English) EPABF1 20374
Total word count - document A 22305
Total word count - document B 0
Total word count - documents A + B 22305

...SPECIFICATION The kuseg 150 segment for the kernel mode is 2-GByte in size, coincident with the "kuseg" of the user mode, so when in the **kernel** mode the processor treats references to this segment just like user mode references, thus **streamlining** kernel access to user data. The kuseg 150 is used to **hold** user code and data, but the operating system often needs to reference this same code or data. The kseg0 area 152 is a 512-MByte...

...This kseg2 area differs from the kuseg area 150 only in that it is not accessible in the user mode, but instead only in the **kernel mode**. The operating system uses kseg2 for stacks and per-process data that must remap on **context switches**, for user page tables (memory map), and for some dynamically-allocated data areas. Kseg2 allows selective caching and mapping on a per page basis, rather...

18/3,K/18 (Item 18 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00317077
Multi-mode control of virtually-addressed unified cache.
Steuerung mit mehrfachen Betriebsarten fur vereinheitlichten und virtuell adressierten Cachespeicher.
Commande multimode d'antememoire unifiee et adressee virtuellement.
PATENT ASSIGNEE:
COMPUTER CONSOLES INCORPORATED (a Delaware corporation), (1022200), 9801 Muirlands Boulevard, Irvine California 92718-2521, (US), (applicant designated states: DE;FR;GB;IT)
INVENTOR:
Sims, Roger Scott, 22945 Springwater Street, E1 Toro California 92630, (US)
Benkual, Jack, 25906 Greenbank, E1 Toro California 92630, (US)
Slingwine, John David, 26381 Via Conchita, Mission Viejo California 92691 , (US)
LEGAL REPRESENTATIVE:
Meddle, Alan Leonard et al (33761), FORRESTER & BOEHMERT Widenmayerstrasse 4/I, D-8000 Munchen 22, (DE)
PATENT (CC, No, Kind, Date): EP 310444 A2 890405 (Basic)
EP 310444 A3 900613
APPLICATION (CC, No, Date): EP 88309149 880930;
PRIORITY (CC, No, Date): US 104352 871002
DESIGNATED STATES: DE; FR; GB; IT
INTERNATIONAL PATENT CLASS (V7): G06F-012/10;
ABSTRACT WORD COUNT: 141

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS A (English) EPABF1 286
SPEC A (English) EPABF1 5832
Total word count - document A 6118
Total word count - document B 0
Total word count - documents A + B 6118

...SPECIFICATION This is for a separate cache for instruction buffering. It is organized and implemented differently from the unified cache. It is organized as a 4-**block**, 1-set associative cache for storing eight **instructions** in each of its four locations. It is implemented in static RAM cells for maximum speed. In contrast, the unified cache is implemented in dynamic...

...of the same properties as a micro routine in a microcoded machine. Micro mode is normally entered as a result of execution of a predetermined **instruction** in the instruction repertory (an MCALL

instruction), or **event**.

User mode and **kernel mode** are similar to the nonprivileged and **privileged modes** of most conventional processors. User mode is used to execute application code, kernel mode to execute the majority of the kernel.

IP 101 has a...the Special Cache is used to cache instructions and data needed to manage TC loads. For both the IB cache and the unified cache, cache **block** size (or location width) is 64 bytes/8 **instructions**/16 PTEs. The IB has 1x4 cache **blocks**; the IDC has 64x2 cache **blocks**; the TC and SC each have 8x2 **blocks**. During execution of a store **instruction**, data can be stored into a location in the IDC. This is not the case for the IB, TC, and SC.

The IDC, TC, and...calculate the address of and load the PTE mapping the virtual page number; restore registers saved above; exit from T-mode by executing an MRET **instruction**.

Loading the PTE mapping the virtual address causes the **block** of 16 PTEs containing the PTE to be loaded into the translation cache. This requires that PTEs be stored in the first 256 megabytes of...

18/3,K/19 (Item 19 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00313967
Apparatus and method for providing an extended processing environment on nonmicrocoded data processing system
Vorrichtung und Verfahren zur Versorgung einer erweiterten Verarbeitungsumgebung fur ein nichtmikrokodiertes Datenverarbeitungssys tem
Dispositif et methode pour fournir un environnement de traitement etendu pour un systeme non microcode du traitement des donnees
PATENT ASSIGNEE:
DIGITAL EQUIPMENT CORPORATION, (313081), 111 Powdermill Road, Maynard Massachusetts 01754-1418, (US), (applicant designated states: CH;DE;FR;GB;IT;LI;NL;SE)
INVENTOR:
Cutler, David N., 3451 Evergreen Road, Bellevue Washington 98004, (US)
Orbits, David A., 17028 North East 139th Street, Redmond Washington 98052 , (US)
Bhandarkar, Dileep, 3 Lantern Lane, Shrewsbury Massachusetts 01545, (US)
Cardoza, Wayne, 3 Hutchinson Road, Merrimack New Hampshire 03054, (US)
Witek, Richard T., 3 Silver Birch Lane, Littleton Massachusetts 01460, (US)
LEGAL REPRESENTATIVE:
Goodman, Christopher et al (31122), Eric Potter & Clarkson St. Mary's Court St. Mary's Gate, Nottingham NG1 1LE, (GB)
PATENT (CC, No, Kind, Date): EP 301707 A2 890201 (Basic)
EP 301707 A3 920108

EP 301707 B1 960117
APPLICATION (CC, No, Date): EP 88305993 880630;
PRIORITY (CC, No, Date): US 69365 870701
DESIGNATED STATES: CH; DE; FR; GB; IT; LI; NL; SE
INTERNATIONAL PATENT CLASS (V7): G06F-009/44; G06F-009/46;
ABSTRACT WORD COUNT: 150

LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS A (English) EPABF1 820
CLAIMS B (English) EPAB96 1066
CLAIMS B (German) EPAB96 993
CLAIMS B (French) EPAB96 1226
SPEC A (English) EPABF1 3937
SPEC B (English) EPAB96 4071
Total word count - document A 4757
Total word count - document B 7356
Total word count - documents A + B 12113

...ABSTRACT sequences as a atomic operation, the nonmicrocoded data processing system is provided with a third mode, in addition to the (nonprivileged) user mode and the (**privileged**) **kernel mode**, of operation that permits the execution of **instruction** sequences with interrupting **events** disabled and certain functions and apparatus enabled to facilitate instruction sequence execution.

...SPECIFICATION and nonmicrocoded data processing systems.
It is yet another feature of the present invention to provide a data processing system operating mode for execution of **instructions** in addition to the usual nonprivileged mode and **privileged mode**.

It is still another feature of the present invention to permit execution of preselected groups of instructions in an atomic manner without interruption.
It is...

...THE INVENTION
The aforementioned and other features are accomplished, according to the present invention, by providing a data processing system with three modes for executing **instruction**. The user mode, in which application programs are generally executed, and the **kernel mode**, in which privileged **instructions** are typically executed, are the usual modes of a data processing system. To these modes, the present invention adds a third mode, hereinafter referred to
...

...is a diagrammatic illustration of the relationship of the data processing system operating modes.

Figure 5 illustrates the steps in transferring from the user or **kernel mode** to the EPICODE mode.

Figure 6 illustrates that the EPICODE **instruction** sequences are stored in reserved areas of the main memory unit.

DESCRIPTION OF THE PREFERRED EMBODIMENT

1. Detailed Description of the Figures

Referring now to...control in order to obtain the desired processing capabilities. The instructions are typically nonprivileged in the sense that the order and selected aspects of the **instruction** are under control of the user. The **kernel mode** 4B is the mode in which the operating system executes **instructions**. The **kernel mode** executes all **instructions** available in the user mode as well as additional **instructions** associated with the **kernel mode** 4B that are privileged and therefore are not available for manipulation by a user. Privileged **instructions** are not allowed in user mode because they could compromise the security of other users or programs. The EPICODE mode of data processing system operation...

...reserved for instruction sequences that should execute without interruption and/or should not execute unless the data processing system is in a predetermined state. Some **instructions** that can be executed in user mode 4A or in **kernel mode** 4B require a transition into the EPICODE mode 4C. This mode is provided with certain privileges and certain dedicated hardware implementing the strategy to ensure...

...instruction in the EPICODE format communicates to the data processing system the requirement to enter the EPICODE mode. In step 502, the issue unit is **prevented** from issuing new **instructions**, but the **instructions** for which execution has begun are completed. The completion of currently executing instructions permits all hardware exceptions to be signaled prior to execution in the...expedited, there being no need to transfer individual data and instruction elements. In addition, programs are generally written in a format that stores data and **instruction** elements needed for sequential **instruction** execution relatively **close** together in the program or file. Thus, a page of data and **instruction** elements will typically include a multiplicity of related data and instruction elements for program execution. None-the-less, the relative rigidity of the granularity of...

...SPECIFICATION and nonmicrocoded data processing systems.

It is yet another feature of the present invention to provide a data processing system operating mode for execution of **instructions** in addition to the usual nonprivileged mode and **privileged mode**

It is still another feature of the present invention to permit execution of preselected groups of instructions in an atomic manner without interruption.

It is...

...THE INVENTION

The aforementioned and other features are accomplished, according to the present invention, by providing a data processing system with three modes for executing **instruction**. The user mode, in which application programs are generally executed, and the **kernel mode**, in which privileged **instructions** are typically executed, are the usual modes of a data processing system. To these modes, the present invention adds a third mode, hereinafter referred to ...

...is a diagrammatic illustration of the relationship of the data processing system operating modes.

Figure 5 illustrates the steps in transferring from the user or **kernel mode** to the EPICODE mode.

Figure 6 illustrates that the EPICODE **instruction** sequences are stored in reserved areas of the main memory unit.

DESCRIPTION OF THE PREFERRED EMBODIMENT

1. Detailed Description ...control in order to obtain the desired processing capabilities. The instructions are typically nonprivileged in the sense that the order and selected aspects of the **instruction** are under control of the user. The **kernel mode** 4B is the mode in which the operating system executes **instructions**. The **kernel mode** executes all **instructions** available in the user mode as well as additional **instructions** associated with the **kernel mode** 4B that are privileged and therefore are not available for manipulation by a user. Privileged **instructions** are not allowed in user mode because they could compromise the security of other users or programs. The EPICODE mode of data processing system operation...

...reserved for instruction sequences that should execute without interruption and/or should not execute unless the data processing system is in a predetermined state. Some **instructions** that can be executed in user mode 4A or in **kernel mode** 4B require a transition into the EPICODE mode 4C. This mode is provided with certain privileges and certain dedicated hardware implementing the strategy to ensure...

...instruction in the EPICODE format communicates to the data processing system the requirement to enter the EPICODE mode. In step 502, the issue unit is **prevented** from issuing new **instructions**, but the **instructions** for which execution has begun are completed. The completion of currently executing instructions permits all hardware exceptions to be signaled prior to execution in the...expedited, there being no need to transfer individual data and instruction elements. In addition, programs are generally written in a format that stores data and **instruction** elements needed for sequential **instruction**

execution relatively **close** together in the program or file. Thus, a page of data and **instruction** elements will typically include a multiplicity of related data and instruction elements for program execution. None-the-less, the relative rigidity of the granularity of ...

...CLAIMS B1

1. A data processor (11), which executes **instructions** in a user mode (4A) of operation and a **kernel mode** (4B) of operation, for processing programs and for servicing interrupt requests that are applied to said processor, characterized in that said data processor comprises:
processing...

...wherein the control means (22) enables the processing means to execute non-privileged instructions when operating in the user mode (4A), privileged or non-privileged **instructions** when operating in the **kernel mode** (4B), and non-interruptible sequences of **instructions** when operating in the third mode (4C).

3. The data processor of Claim 2 wherein said non-interruptible sequences of instructions include special instructions in...

...in said third mode.

17. The data processor of Claim 1 wherein said control means enables said processing means to execute a plurality of overlapping **instructions** in said user mode and said **kernel mode**, said control means enabling said processing means to complete execution (502) of all currently executing **instructions** in said user and **kernel mode** prior to entering said third mode.

18. A method of operating a data processor (11), which executes **instructions** in a user mode (4A) of operation and a **kernel mode** (4B) of operation, characterized in that said method comprises the steps of:
enabling said processor to perform processing operations in at least three privilege modes...not include microcode for performing said operations that are processed in said third mode.

21. A method of operating a data processor (11) which executes **instructions** in a user mode (4A) of operation and a **kernel mode** (4B) of operation, characterized in that said method comprises the steps of:
allowing said processor to temporarily suspend executing said **instructions** and service interrupt requests when said processor is operating in said user mode or said **kernel mode**; and

causing said processor to respond to a predetermined **instruction** executed in said user mode by executing a selected sequence of instructions (509), and inhibiting (503) said processor from suspending execution of said sequence and...

18/3,K/20 (Item 20 from file: 348)
DIALOG(R)File 348:EUROPEAN PATENTS

00281658
MULTIPROCESSING METHOD AND ARRANGEMENT.
MULTIPROZESSORVERFAHREN UND -ANORDNUNG.
PROCEDE ET AGENCEMENT MULTIPROCESSEURS.
PATENT ASSIGNEE:
AMERICAN TELEPHONE AND TELEGRAPH COMPANY, (589370), 550 Madison Avenue, New York, NY 10022, (US), (applicant designated states: DE;FR;GB;IT;NL)
INVENTOR:
STRELIOFF, Brian, Kent, 4683 Apt. 1B Lake Valley, Lisle, IL 60532, (US)
LEGAL REPRESENTATIVE:
Watts, Christopher Malcolm Kelway, Dr. et al (37392), AT&T (UK) LTD. AT&T Intellectual Property Division 5 Mornington Road, Woodford Green Essex IG8 OTU, (GB)
PATENT (CC, No, Kind, Date): EP 354899 A1 900221 (Basic)
EP 354899 B1 930303
WO 8805943 880811
APPLICATION (CC, No, Date): EP 87905127 870727; WO 87US1802 870727
PRIORITY (CC, No, Date): US 12085 870206
DESIGNATED STATES: DE; FR; GB; IT; NL
INTERNATIONAL PATENT CLASS (V7): G06F-009/46;
NOTE:
No A-document published by EPO
LANGUAGE (Publication,Procedural,Application): English; English; English
FULLTEXT AVAILABILITY:
Available Text Language Update Word Count
CLAIMS B (English) EPBBF1 1612
CLAIMS B (German) EPBBF1 1556
CLAIMS B (French) EPBBF1 1777
SPEC B (English) EPBBF1 8387
Total word count - document A 0
Total word count - document B 13332
Total word count - documents A + B 13332

...SPECIFICATION the invention.
Typically, the 3B15 system uses two modes, or levels, of operation of the WE 32100 microprocessor: a user mode for executing user program **instructions**, and a **privileged mode** for executing functions such as operating system **instructions** that have the potential for corrupting shared system resources such as hardware resources. There are two mechanisms for entering **privileged mode** from user mode: a process switch mechanism and a **system call** mechanism. The **system call** mechanism is also known by names such as a supervisory call and an operating system trap.

The system call effectively acts as a subroutine call...

...provides a means of controlled entry into a function by installing on a processor a new processor status word (PSW) and program counter (PC). The **system call** mechanism is used by explicit operating **system** GATE **calls** to transfer control from user to **privileged mode**, and is also used to handle "normal" system exceptions. ("Quick" interrupts, which would also be handled by this mechanism, are not used by the UNIX...

...exception is an error condition--a fault or a trap--other than an interrupt. Normal exceptions constitute most system exceptions. The normal exception handlers are **privileged-mode** functions.

The **system call** mechanism uses the execution stack of the present process; that is, a normal exception handler or a function called via a GATE instruction uses for...

...privileged state will not become corrupted. Therefore, prior to making a legal entry of a privileged function, i.e., before executing a transfer to the **privileged mode** upon the occurrence of a normal exception or a GATE request, the **system call** mechanism checks the present execution stack pointer value against the execution stack boundary values that are stored in the process control block of the presently...processor 12, or execution of the partially-executed instruction is merely completed on master processor 12. Unless the condition that caused the attempt to enter **privileged mode** was a transient fault, execution of that **instruction** on master processor 12 results in the invocation of the operating system service. That service is then provided in a conventional, uniprocessor, manner on master...the physical address of the slave's initial process control block). Also at step 502, a process switch to the slave's initial process control **block** is performed by the conventional RETPS **instruction**. As part of this process switch, the slave's process control **block** pointer register is set to the physical address of the intermediate portion of the slave's initial process control block, the initial program counter is...

...step 503, which resets the process control block pointer register to the virtual address of the intermediate portion of the slave's initial process control **block**, and executes the conventional ENBVJMP **instruction** which enables virtual addressing for slave processor 25 and transfers to the virtual address contained in register r0.

The slave virtual-mode startup routine is...This is implemented by calling a slave delete routine (see FIG. 15) , at step 1002. Illustratively in this example, because the address of the faulting **instruction** is already saved in the process control **block** as result of stack exception handling, the faulting **instruction** is reexecuted once the process restarts execution on master processor 12. This feature is fairly typical of systems with demand-paged memory

management. This results...

...routine (see FIG. 14) must be invoked to avoid restart problems. This is illustratively accomplished, at step 1001, by removing the address of the faulting **instruction** from the process control **block** and substituting therefor the starting address of the **instruction** restart routine, and storing the removed faulting instruction's address in a variable. When the **instruction** restart routine completes, it restores the faulting **instruction**'s address in the process control **block**.

Although far less usual than normal exceptions, another exception condition is possible when executing user-mode processes: system error. This category includes things like alignment...

...as the standard, master, stack exception handler deals with MAU restart problems. The instruction restart routine then restores the address of the user process' faulted **instruction** to the process control **block** of the faulted process to cause execution of the faulted **instruction**, at step 1203, and returns at step 1204 to execution of that instruction.

FIG. 15 flowcharts the slave delete routine. This routine is invoked on ...

...CLAIMS indicator means associated with the occurred event.

12. The system of claim 11 in a system wherein a change of execution mode comprises entry of **privileged** execution **mode**: wherein
the plurality of **events** comprise interrupts; wherein
the indicators comprise interrupt vectors: and wherein
the functions identified by the indicators comprise interrupt handlers executable in privileged mode.

13. The...

18/3,K/27 (Item 27 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

01029412
METHODS AND SYSTEM FOR MANAGING COMPUTATIONAL RESOURCES OF A COPROCESSOR IN A COMPUTING SYSTEM
Patent Applicant/Assignee:
MICROSOFT CORPORATION, One Microsoft Way, Redmond, WA 98053, US, US (Residence), US (Nationality)
NENE Sameer A, 15816 N.E. 46th Ct., Redmond, WA 98052, US, US (Residence), US (Nationality)
BEDA Joseph S III, 3819 Densmore Ave. N., Seattle, WA 98103, US, US (Residence), US (Nationality)

Inventor(s):
WILT Nicolas P, 1920 - 205th Place Northeast, Sammamish, WA 98074, US,
Legal Representative:
ROCCI Steven J (et al) (agent), Woodcock Washburn LLP, 46th Floor, One Liberty Place, Philadelphia, PA 19103, US,
Patent and Priority Information (Country, Number, Date):
Patent: WO 200358431 A1 20030717 (WO 0358431)
Application: WO 2003US396 20030106 (PCT/WO US0300396)
Priority Application: US 200239036 20020104
Publication Language: English
Filing Language: English
Fulltext Word Count: 20019

Patent and Priority Information (Country, Number, Date):
Patent: ...20030717
Fulltext Availability:
Detailed Description
Claims
Publication Year: 2003

Detailed Description
... primitive is an object that can be used to synchronize multiple threads' access to shared resources, such as critical sections, mutexes, semaphores or events.

A **thread** is an executable entity that comprises a program counter, a user-mode stack, a **kernel-mode** stack and a set of register values.

A token **stream** is a **stream** of hardware-independent tokens that describe a series of drawing operations. A token stream can be translated by a hardware-specific software component, such as... inefficient and generally does not scale well with multiple instances of a given piece of hardware in a system. As a rule, IN and OUT **instructions** can be executed only in **kernel mode**, the microprocessor mode that allows direct manipulation of hardware. If a user mode **thread** encounters ...application activity into DrawPrimitives2 ("DP2") tokens. When the DP2 token stream is submitted, a kernel transition occurs and the driver 314 translates the DP2 token **stream** into hardware-specific commands in **kernel mode**.
Fig. 3B does not make any assumptions about whether the driver 314 is in user mode or kernel mode, and in this regard, driver component...that need to be written to this buffer, however, manually tracking this byte count WO 03/058431 PCT/US03/00396
3 1
surround the write **instructions** with a structured exception handling **block**, field the exception and return an error to the runtime if the command buffer overflows. The interface to the user mode driver would then be...commands' DDI counterparts into hardware-specific

commands and write them into a temporary buffer or buffers (since the exact size of the hardware-specific command **stream** is not known until after the translation has been performed). When the recording is **stopped**, or once the token **stream** has been parsed, the system could then allocate a command buffer or command buffers of suitable size and copy the translated hardware commands into them...could be used. The system of Fig. 7 closely resembles the state of the art in DIRECT3D(V driver structure, in that a DrawPrimitives2 token **stream** is passed to the **kernel mode** driver 705. The main difference is that the system of Fig. 7 contemplates OS-arbitrated scheduling of the command buffers, while DIRECT3D@ currently does not...

Claim
... including preempting by the at least one coprocessor upon the occurrence of an external event.

19 A method according to claim 18, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the ...at least one coprocessor upon the occurrence of an external event.

45 At least one computer readable medium according to claim 44, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the at least one coprocessor.

46 At least one computer readable medium according to claim 27, wherein the host processor is...preempted by the at least one coprocessor upon the occurrence of an external event.

70 A computing device according to claim 69, wherein the external **event** is the operating **system** making a **call** to a corresponding **kernel mode** driver object to preempt the at least one coprocessor.

71 A computing device according to claim 52, wherein the host processor is interrupted to coordinate...

18/3,K/28 (Item 28 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

01000985
GENERAL PURPOSE FIXED INSTRUCTION SET (FIS) BIT-SLICE FEEDBACK PROCESSOR UNIT/COMPUTER SYSTEM

Patent Applicant/Inventor:

COOPER Benjamin, 39204 Opalocka Rd., Boulevard, CA 91905, US, US (Residence), -- (Nationality)

Patent and Priority Information (Country, Number, Date):
Patent: WO 200329960 A1 20030410 (WO 0329960)
Application: WO 2002US29534 20020916 (PCT/WO US0229534)
Priority Application: US 2001326170 20011001
Parent Application/Grant:
Related by Continuation to: US 2001326170 20011001 (CIP)
Publication Language: English
Filing Language: English
Fulltext Word Count: 51071

Patent and Priority Information (Country, Number, Date):
Patent: ...20030410
Fulltext Availability:
Detailed Description
Claims
Publication Year: 2003

Detailed Description
... That is, the computer "freezes" up. Or to put it another way, the computer enters into an infinite loop. When this happens the computer system **stops** responding to the user's **instructions** and input.

Then there are those arrows that are to be found ...data, in this case addressing values, travels from the "general purpose FIS Processor Unit" to the "Rest of general purpose FIS Computer." When one examines **closely** this particular data **stream**, what one finds is that ...again to return to the use of the Kernel Addressing Systeni-retuming the general purpose FIS processor unit of claims (2) and (6) to the **Kernel Mode**.

The second **event** that the "Fundamental Control Memory System" carries out is one that occurs whenever the system is going from the Kernel Addressing System to the Application...system has control of the "general purpose FIS Processor Unit", it then can make a determination. It can either allow the process that was 'ust **stopped** to run again for a given number of **instructions**. Or it can then allow another application, another process, to have its tam at directing this said 11general purpose FIS Processor Unit." That is, the into the **Kernel Mode** after completing a certain number of **instructions**, is the most crucial of them.

Function Four
In the present processors built upon logic circuits, there are a large number of registers in the...program will, in general, operate in a multitasking environment or in a single application environment.
Multitasking and Mimicry
And as stated in claim (27), this **instruction** set that will be

applicable in both of these modes, the **Kernel Mode** and the Application Mode, can also be designed to serve another purpose; that of fitting as **closely** as possible with the **instruction** set, or **instruction** sets, that go along with the more popular, presently manufactured general purpose FIS microprocessors. hi doing this latter task, that of mimicking a present general...feedback Programmed Memory Systems". In particular, that group of "Primary bit-slice feedback Programmed Memory Systems" will be set up so as to mimic as **closely** as possible-and in some cases exactly-the various **instruction** sets of the various types of presently manufactured general purpose FIS microprocessors, as stated in claim (39).

And to do this mimicry, it will be...on the first of the three data transfer subsystems of the "Data Input/Output Bus", two actions take place.

The first is that the the "**Hold**" subsystem shown in fig. 2 stores the **instruction** value. The second
54 of 70
action is that the same **instruction**, now held by the "**Hold**" subsystem, is passed through the multiplexer ...a Data RAM system and the addressing value for that said Data RAM system has stepped forward by one. And what now happens in a **block** integer adder **instruction** is that the value stored in the above mentioned **hold** register/small memory circuit is reduced by one.

Then if the value in the hold register/small memory circuit has reached the set ...integer addition can take place.

And thus in this way the master controller carries out as many additions as the value first received by the **hold** register/small memory circuit when this **block** integer addition **instruction** was first begun.

Chatige Addressingfor RAM
The next function that the master controller must be able to accomplish is that of setting up addressing values...be sent to this new type of general processor. - Or put another way, any other type of move instruction other than the in-page move **instructions** will interfere with the smooth execution of a given application program. However in the **kernel mode**, the processor can change any of the addressing values for any of the four sets of addressing/accessing subsystems for any of the RAM or...must be noted that the use of this Twos Complement Unit, like the use of the integer Adder, can be run as part of a **block instruction**; that is, a whole sequence of numbers can be converted to their twos complement in one long sequence, a sequence that will be coordinated out...to step forward by one and then send the next instruction to itself. At which point the master controller goes on to execute this next **instruction**.

The master controller will also be able to do **block** ANDs, ORs or XORs.

Claim
... be put to is to allow the general purpose FIS processor unit of claims (2) and (6) to keep track of how may times a **block** addition, **block** multiplication, **block** move and the like is repeated in a given **instruction**.

63 that the various types of general purpose FIS processor units of claims (2) and 6) can be so designed so that two or more...

18/3,K/29 (Item 29 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00523481 **Image available**
DYNAMICALLY CONFIGURABLE DATA STORAGE AND PROCESSING SYSTEM OPTIMIZED FOR PERFORMING DATABASE OPERATIONS

Patent Applicant/Assignee:
RECURSION DYNAMICS INC,
GELMAN Boris,
KUMETS Alex,
Inventor(s):
GELMAN Boris,
KUMETS Alex,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9954833 A2 **19991028**
Application: WO 99US8318 19990415 (PCT/WO US9908318)
Priority Application: US 9863085 19980420
Publication Language: English
Fulltext Word Count: 21440

Patent and Priority Information (Country, Number, Date):
Patent: ...19991028
Fulltext Availability:
Detailed Description
Claims
Publication Year: 1999

Detailed Description
... the database software engine is responsible for execution of query plans generated by a query management portion of the database engine. The typical number of **instructions** per data unit for the data manipulation portion is **close** to 1. As a result, a CPU capable of performing 15 (or 1 0 with SRAM) instructions in the time it takes to

fetch each...manipulation portion of the RDBMS. The DMS provides highly efficient execution of all data manipulation operations required for database processing upon multiple, high-bandwidth data **streams** flowing directly from a set of external storage devices which **hold** one or more databases. The DMS is attached directly to the external storage devices.

A preferred embodiment of the architecture of the present invention is... 1604 via a one-way data manipulation instruction signal path and a bi-directional control signal path. Note that in Fig. 16 single lines between **blocks** represent data manipulation **instruction** signal paths, double lines represent data flow paths, and dotted lines represent control signal paths. In all cases, the arrow heads represent the direction of...

...1704 via a one-way data manipulation instruction signal path and a bi-directional control signal path. Note that in Fig. 17 single lines between **blocks** represent data manipulation **instruction** signal paths, double lines represent data flow paths, and dotted lines represent control signal paths. In all cases, the arrow heads represent the direction of...binaries are SP 1406, 1408, 1410, 1412 independent and can be dynamically assigned to any SP 14069 1408@ 1410@ 1412 of any DMP 1452.

Nano-**thread** **instructions** run in either of two modes: user execution mode or **kernel** 1 5 **mode**. All user mode **instructions** are executed by SPs 1406, 1408, 1410, 1412 without any assistance from any other DMP 1452 components. **Kernel** **mode** **instructions** are first intercepted by the CU 1404 and then either executed in the CU 1404 or redirected to the BMI 1416 or the GPI 1402 for execution. **Kernel** **mode** **instructions** allow the DMP 1452 to dynamically reconfigure the SPs 1406, 1408, 1410, 1412 to process different Nanothreads. This allows the DMP 1452 to perform **thread** switching so as to utilize all available resources at any given time. In addition, the **kernel** **mode** **instructions** allow the DMP 1452 ...signal ports for connecting to the CU 1704 of Fig. 17. The LCUCR 191 0 communicates with the CU and delivers all control signals and **kernel** **mode** **instructions** back and forth between the CU and the SP 1900.

Internal control information that is necessary to support its own operation is stored in the...Turning to Fig. 20, a flowchart depicting the interaction between the SP and the CU is presented. In Step SI, the SP waits for a **kernel** **mode** **instruction** and a control signal from the CU. In Step S2, the SP has received from the CU a **kernel** **mode** **instruction** defining a nano-**thread** address in the RFM and a control signal, and as a result the SP switches to the user execution mode from the wait mode. The SP proceeds with nano-thread execution. In Step S3, the SP fetches a nano-thread **instruction**. In Step S4, the instruction is decoded. Nano-thread

execution may be stopped for two reasons: (a) the last **instruction** of a nano-**thread** is reached; or (b) a **kernel mode instruction** is reached. If the **instruction** is a **kernel mode instruction**, a control signal is issued to the CU in Step S5, the **kernel mode instruction** is sent to the CU, and the SP switches to wait mode in Step S7 before returning to Step S 1. The SP remains in the wait mode until it receives new **kernel mode instruction** from the CU. If the **instruction** decoded in Step S4 is a **STOP instruction** indicating the last **instruction** in the thread has been reached, a control signal is issued to the CU in Step S8 and the SP switches to wait mode in Step S7 before returning to Step SI. Again, the SP remains in the wait mode until it receives new **kernel mode instruction** from the CU. If the **instruction** decoded in Step S4 is a data processing **instruction**, the SP moves to Step S9 where the instruction is executed and then it returns to Step S3 to fetch another instruction.

Turning to Fig. 21, the operation of the DMP 2100 is depicted. The ALU SPs 2106 run independent nano-threads concurrently. Running in user mode, nano-**threads** execute data processing **instructions** that operate with the RFM 2114. Each nano-thread can issue **kernel mode instructions** for BAM and RAM access, I/O, communication with the other DMSs and the GPC, or **thread** manipulation. All **kernel mode instructions** are handled by the CU 2104, and then either executed by CU 2104 as **thread** manipulation or redirected for execution on the BMI 2116 or the GPI 2102. The CU 2104 handles and executes multiple **kernel mode instructions** concurrently and supports parallel operation of 43all DMP 21 00 components. The BMI 2116 executes BAM access instructions received from the CU 2104. It performs...

Claim
... to the means for
instruction decoding; and
means for control signal handling and manipulating in response to the instruction execution. . A method of processing data **streams** in a sub-processor comprising the steps of(a) waiting for a kernel **mode instruction** and a control signal from a control unit (b) receiving a **kernel mode instruction** pointing to an address in a register file
memory containing a nano-**thread** for execution and a control signal;
(c) switching to a user execution mode from a wait mode;
(d) fetching the nano-**thread instruction** pointed to by the **kernel mode instruction**
or the next unexecuted nano-**thread instruction**;
(e) decoding the fetched **instruction**;
(f) halting execution of the nano-**thread** if the decoded instruction is a **kernel mode instruction**;

(g) halting execution if the decoded **instruction** is a last **instruction** in the nano **thread**; and
(h) executing the nano-thread instruction if the instruction is a data processing instruction and then returning to step (d) to fetch the next
...

...nano-thread instruction.

59 The method of claim 58 wherein the step (f) finther comprises:
sending a control signal to the control unit;
sending the **kernel mode instruction** to the control unit;
switching to the wait mode from the user execution mode; and
returning to step (a).

60 The method of claim 58 comprising the steps of.
switching to a user mode from a **kernel mode** in response to a **kernel mode instruction**;
executing nano-**thread** data processing **instructions** concurrently on one or more sub-processors;
storing the results of nano-**thread** data processing **instructions** in a register file memory;
switching to the **kernel mode** from the user mode in response to either reaching a I 0 **kernel mode instruction** while executing the nano-**thread** or reaching the end of the nano thread;
executing, on a control unit, **kernel mode instructions** received from the sub processors as **thread** manipulations;
redirecting, by the control unit, **kernel mode instructions** to a burst mode interface 1 5 or a general purpose interface;
executing, on the burst mode interface, **kernel mode instructions** to access burst mode memory and writes to the register file memory;
executing, on the general purpose interface, **kernel mode instructions** to perform random access memory accesses, input/output **instructions**, and communication **instructions** received from the control unit, wherein execution of the instructions involve data transfers between the register file memory and a memory controller, an input/output adapter, a system bus adapter, or a data manipulation system controller ; and redirecting, by the general purpose interface, **kernel mode instructions** to the memory controller, the input/output adapter, the system bus adapter, or the data manipulation system controller. . A method of performing data manipulation functions

...

18/3,K/30 (Item 30 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00523461 **Image available**
RISC PROCESSOR WITH CONTEXT SWITCH REGISTER SETS ACCESSIBLE BY EXTERNALCOPROCESSOR

Patent Applicant/Assignee:
TRANSWITCH CORP,
Inventor(s):
ROY Subhash C,
HEMBROOK Paul,
PARRELLA Eugene L,
MARIANO Richard,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9954813 A1 **19991028**
Application: WO 99US8275 19990414 (PCT/WO US9908275)
Priority Application: US 9864446 19980422
Publication Language: English
Fulltext Word Count: 7546

Patent and Priority Information (Country, Number, Date):
Patent: ...19991028
Fulltext Availability:
Detailed Description
Publication Year: 1999

Detailed Description
... the kernel context may occur frequently. According to the MIPS ISA, the CPU enters the kernel mode whenever an exception is detected and remains in **kernel mode** until a Restore From Exception (RFE) **instruction** is executed. Consequently, in an **event** driven application, frequent context switches can be expected regardless of the number of threads in user modes.

The relative high speed of RISC processors make...to the detailed description taken in conjunction with the provided figures.

BRIEF DESCRIPTION OF THE DRAWINGS
Figure 1 is a diagram of prior art pipeline **instruction** processing in a MIPS processor; Figure 2 is a schematic **block** diagram of the major functional blocks of a processor
according to the invention;
Figure 3 is a schematic block diagram of the major functional blocks...a Con-sel code provides many significant advantages, particularly for real

time event-driven applications.

For example, during interrupt processing, when a MIPS processor normally **switches context** from user mode to **kernel mode**, the processor according to the invention need not save and restore register contents. The processor according to the invention can switch to **kernel mode** in three **instruction** cycles and back to user mode in another three **instruction** cycles. Further, more than two threads are rapidly supported by loading register contents in the background via the AXI port with a coprocessor. The provision...the epc to the pc. The program should first move the epc to a general purpose register and exit the exception handler using a JR **instruction**.

The present invention also adds a second status register to the interface **block** 78. The additional register, status txc, accommodates the additional coprocessors and masks for additional interrupt signals. This leaves the MIPS status register unaltered and preserves

18/3,K/31 (Item 31 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00489737
METHOD AND APPARATUS FOR ALTERING THREAD PRIORITIES IN A MULTITHREADED PROCESSOR
Patent Applicant/Assignee:
INTERNATIONAL BUSINESS MACHINES CORPORATION,
Inventor(s):
BORKENHAGEN John Michael,
FLYNN William Thomas,
WOTTRENG Andrew Henry,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9921089 A1 **19990429**
Application: WO 98US21724 19981014 (PCT/WO US9821724)
Priority Application: US 97958718 19971023
Publication Language: English
Fulltext Word Count: 13620

Patent and Priority Information (Country, Number, Date):
Patent: ...19990429
Fulltext Availability:
Detailed Description
Publication Year: 1999

Detailed Description
... cache level which can be completed in much less time.

Thrashing, wherein each thread is locked in a repetitive cycle of switching threads without any **instructions** executing, can be **prevented** by the invention implementing a forward progress count register and method which allows up to a programmable maximum number of thread switches after which the...processing system according to the present invention.

Figure 3 illustrates a block diagram of the storage control unit of Figure 2.
Figure 4 illustrates a **block** diagram of the thread switch logic, the storage control unit and the **instruction** unit of Figure 2.

Figure 5 illustrate the changes of state of a thread as the thread experiences different thread switch events shown in Figure...which
allows up to a programmable maximum number of thread switches called the forward progress threshold value.

After that maximum number of thread switches, an **instruction** must be completed before switching can occur again. In this way, thrashing is **prevented**. Forward progress count register 420 may actually be bits 30:31 in the thread switch control register 410 or a software programmable forward progress threshold...

...610, bits 15:17 in thread state
register 442 pertaining to thread TO are reset to state 111. Execution of this thread is attempted in **block** 620 and the state changes to 000. If an **instruction** successfully executes on thread TO, the state of thread TO returns to 111 and remains so. If, however, thread TO cannot execute an instruction, a...itself to change the priority of itself.

tsop I or 1,1,1 - Switch to dormant thread
tsop 2 or 1,1,1 - Set active **thread** to LOW priority
- Switch to dormant **thread**
- NOTE: Only valid in **privileged mode** unless TSC[191=1
tsop 3 or 2,2,2 - Set active **thread** to MEDIUM priority
tsop 4 or 3,3,3 - Set active **thread** to HIGH priority
- NOTE: Only valid in **privileged mode**
Priority switch **instructions** tsop 1 and tsop 2 can be the same **instruction** as embodied herein as or 1,1,1 but they can also be separate instructions. These instructions interact with bits 19 and 21 of the...thread switch control register 320 is zero, the

priority for both threads is set to medium and the effect of the or x,x.,x **instructions** on the priority is **blocked**.

If an external interrupt request is active, and if the corresponding thread's priority is low, that thread's priority is set to medium.
The...

18/3,K/32 (Item 32 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00489736 **Image available**
AN APPARATUS AND METHOD TO GUARANTEE FORWARD PROGRESS IN A MULTITHREADED
PROCESSOR
Patent Applicant/Assignee:
INTERNATIONAL BUSINESS MACHINES CORPORATION,
Inventor(s):
BORKENHAGEN John Michael,
EICKEMEYER Richard James,
FLYNN William Thomas,
LEVENSTEIN Sheldon Bernard,
WOTTRENG Andrew Henry,
KUNKEL Steven R,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9921088 A1 **19990429**
Application: WO 98US21715 19981014 (PCT/WO US9821715)
Priority Application: US 97956875 19971023
Publication Language: English
Fulltext Word Count: 13755

Patent and Priority Information (Country, Number, Date):
Patent: ...19990429
Fulltext Availability:
Detailed Description
Claims
Publication Year: 1999

Detailed Description
... processing by the use of a software thread control manager.

Thrashing, wherein each thread is locked in a repetitive cycle of switching threads without any **instructions** executing, can be **prevented** by the invention implementing a forward progress count register and method which allows up to a programmable maximum number of thread switches called the

forward...
...computer networks; and another forward progress threshold for shorter latency events such as cache misses. In these cases, the processor system may use a software **instruction** and the hardware response to an external processor latency event to **block** the incrementing of the forward progress counter in the thread state register, in other words, to block the change of state of the particular thread...or the network by the multithreaded processor causes an external processor latency event. An external latency event causing an excessive processor delay or a software **instruction** can **block** the change of state of the particular thread in the thread state register. Regardless of the state of the particular thread, the multithreaded processor still...

...processing system according to the present invention.

Figure 3 illustrates a block diagram of the storage control unit of Figure 2.

Figure 4 illustrates a **block** diagram of the thread switch logic, the storage control unit and the **instruction** unit of Figure 2.

Figure 5 illustrate the changes of state of a thread as the thread experiences different thread switch events shown in Figure...which allows up to a programmable maximum number of thread switches called the forward progress threshold value. After that maximum number of thread switches, an **instruction** must be completed before switching can occur again. In this way, thrashing is **prevented**. Forward progress count register 420 may actually be bits 30:31 in the thread switch control register 410 or a software programmable forward progress threshold...counter bits 15:17 in thread state register 442 pertaining to thread TO are reset to state 111. Execution of this thread is attempted in **block** 620 and the state changes to 000. If an **instruction** successfully executes on thread TO, the state of thread TO returns to Ill and remains so. If, however, thread TO cannot execute an instruction, a...the executing software itself to change the priority of itself.

tsop I or],],] -Switch to dormant thread
tsop 2 or 1,1,1 -Set active **thread** to LOW priority
-Switch to dormant **thread**
-NOTE: Only valid in **privileged** **mode** unless TSC[19]=l
tsop 3 or 2,2,2 -Set active **thread** to MEDRJM priority
tsop 4 or 3,3,3 -Set active **thread** to HIGH priority

-NOTE: Only valid in **privileged mode**
Instructions tsop 1 and tsop 2 can be the same **instruction** as embodied herein as or 1.1,1 but they can also be separate instructions. These instructions interact with bits 19 and 21 of the...thread
switch control register 320 is zero, the priority for both threads is set to medium and the effect of the or x,,x,,x **instructions** on the priority is **blocked**. If an external interrupt request is active, and if the corresponding thread's priority is low, that thread's priority is set to medium.

The...

Claim
... or the network by the multithreaded processor causes an external processor latency event.

11 The computer system of Claim 9 or 10 further comprising an **instruction** to **block** the change of state of the particular thread in the thread state register.

12 The computer system of any one of Claims 9 to 11...

18/3,K/33 (Item 33 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00489731 **Image available**
THREAD SWITCH CONTROL IN A MULTITHREADED PROCESSOR SYSTEM

Patent Applicant/Assignee:
INTERNATIONAL BUSINESS MACHINES CORPORATION,
Inventor(s):
BORKENHAGEN John Michael,
EICKEMEYER Richard James,
FLYNN William Thomas,
LEVENSTEIN Sheldon Bernard,
WOTTRENG Andrew Henry,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9921083 A1 **19990429**
Application: WO 98US21742 19981014 (PCT/WO US9821742)
Priority Application: US 97957002 19971023
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CA CN CZ HU IL JP KR PL RU AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English

Fulltext Word Count: 15574

Patent and Priority Information (Country, Number, Date):
Patent: ...**19990429**
Fulltext Availability:
Detailed Description
Publication Year: **1999**

Detailed Description
... level but which can be completed in much less time.

Thrashing, wherein each thread is locked in a repetitive cycle of switching threads without any **instructions** executing, can be **prevented** by the invention implementing a progress count register and method which allow up to a programmable maximum number of thread switches af ter which the...processing system according to the present
invention.

Figure 3 illustrates a block diagram of the storage control unit of Figure 2.
Figure 4 illustrates a **block** diagram of the thread switch logic, the storage control unit and the **instruction** unit of Figure 2.

Figure 5 illustrate the changes of state of a thread as the thread experiences different thread switch events shown in Figure...which allows up to
a programmable maximum number of thread switches called the forward progress threshold value. After that maximum number of thread switches, an **instruction** must be completed before switching can occur again. In this way, thrashing is **prevented**. Forward progress count register 420 may actually be bits 30:31 in the thread switch control register 410 or a software programmable forward progress threshold...

...610, bits 15:17 in thread state register 442 pertaining to thread TO are reset to state 111. Execution of this thread is attempted in **block**, 620 and the state changes to 000. If an **instruction** successfully executes on thread TO, the state of thread TO returns to Ill and remains so. If, however, thread TO cannot execute an instruction, a...itself to change the priority of itself.

tsop I or 1,1,1 - Switch to dormant thread
tsop 2 or 1,1,1 - Set active **thread** to LOW priority
- Switch to dormant **thread**

- NOTE: Only valid in **privileged mode** unless TSC[19]=1
tsop 3 or 2,2,2 - Set active **thread** to MEDIUM priority
tsop 4 or 3,3,3 - Set active **thread** to HIGH priority
- NOTE: Only valid in **privileged mode**
Priority switch **instructions** tsop 1 and tsop 2 can be the same **instruction** as embodied herein as or 1,1,1 but they can also be separate instructions. These instructions interact with bits 19 and 21 of the...the thread switch control register 320 is zero, the priority for both threads is set to medium and the effect of the or xx,x **instructions** on the priority is **blocked**. If an external interrupt request is active, and if the corresponding thread's priority is low, that thread's priority is set to medium.

The...

18/3,K/34 (Item 34 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00489730 **Image available**
METHOD AND APPARATUS TO FORCE A THREAD SWITCH IN A MULTITHREADED PROCESSOR
Patent Applicant/Assignee:
INTERNATIONAL BUSINESS MACHINES CORPORATION,
Inventor(s):
BORKENHAGEN John Michael,
EICKEMEYER Richard James,
FLYNN William Thomas,
WOTTRENG Andrew Henry,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9921082 A1 **19990429**
Application: WO 98US21741 19981014 (PCT/WO US9821741)
Priority Application: US 97956577 19971023
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CA CN CZ HU IL JP KR PL RU AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 12699

Patent and Priority Information (Country, Number, Date):
Patent: ...19990429
Fulltext Availability:
Detailed Description
Publication Year: 1999

Detailed Description
... block diagram of the storage
control unit of Fiqure 2.

Figure 4, consisting of two drawing sheets designated
Pigure 4A and Figure 4B, illustrates a **block** diagram of the
thread switch logic, the storage control unit and the
instruction unit of Figure 2.

Figure 5 illustrate the changes of state of a thread as
the thread experiences diff"erent thread switch events shown in...which
allows up to a programmable
maximum number of thread switches called the forward progress
threshold value. After that maximum number of thread switches,
an **instruction** must be completed before switching can occur
again. In this way, thrashing is **prevented**. Forward progress
count register 420 may actually be bits 30:31 in the thread
switch control register 410 or a software programmable forward
progress threshold...

...610, bits 15:17 in thread state register 442
pertaining to thread TO are reset to state 111. Execution of
this thread is attempted in **block** 620 and the state changes to
000. If an **instruction** successfully executes on thread TO, the
state of thread TO returns to Ill and remains so. If, however,
thread TO cannot execute an instruction, a...itself to change the
priority
of itself.

tsop I or 1,1,1-Switch to dormant thread
tsop 2 or 1,1,1-Set active **thread** to LOW priority
-Switch to dormant **thread**
-NOTE: Only valid in **privileged mode** unless TSC[19]=l
tsop 3 or 2,2,2-Set active **thread** to MEDIUM priority
tsop 4 or 3,,3,3-Set active **thread** to HIGH priority
-NOTE: Only valid in **privileged mode**
Instructions tsop I and tsop 2 can be the same **instruction**
as embodied herein as or 1,1,1 but they can also be separate
instructions. These instructions interact with bits 19 and 21
of the...

...thread
switch control register 320 is zero, the priority for both
threads is set to medium and the effect of the or x,x,,x
instructions on the priority is **blocked**. If an external
interrupt request is active, and if the corresponding thread's
priority is low, that thread's priority is set to medium.

The...

18/3,K/35 (Item 35 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00489729 **Image available**
METHOD AND APPARATUS FOR SELECTING THREAD SWITCH EVENTS IN A MULTITHREADED
PROCESSOR
Patent Applicant/Assignee:
INTERNATIONAL BUSINESS MACHINES CORPORATION,
Inventor(s):
BORKENHAGEN John Michael,
EICKEMEYER Richard James,
FLYNN William Thomas,
LEVENSTEIN Sheldon Bernard,
WOTTRENG Andrew Henry,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9921081 A1 19990429
Application: WO 98US21716 19981014 (PCT/WO US9821716)
Priority Application: US 97958716 19971023
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CA CN CZ HU IL JP KR PL RU AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 13400

Fulltext Availability:
Detailed Description
Publication Year: 1999

Detailed Description
... which allows up to a programmable maximum number of thread switches called the forward progress threshold value. After that maximum number of thread switches, an **instruction** must be completed before switching can occur again. In this way, thrashing is **prevented**. Forward progress count register 420 may actually be bits 30:31 in the thread switch control register 410 or a software programmable forward progress threshold...610, bits 15:17 in thread state register 442 pertaining to thread TO are reset to state 111. Execution of this thread is attempted in **block** 620 and the state changes to 000. If an **instruction** successfully executes on thread TO, the state of thread TO returns to 111 and remains so. If, however, thread

. TO cannot execute an instruction, a...itself to change the priority of itself.

tsop I or 1,1,1 -Switch to dormant thread
tsop 2 or 1,1,1 - Set active **thread** to LOW priority
-Switch to dormant **thread**
-NOTE: Only valid in **privileged mode** unless TSC[19]=l
tsop 3 or 2,2,2 -Set active **thread** to MEDRJM priority
tsop 4 or 3,3,3 -Set active **thread** to HIGH priority
-NOTE: Only valid in **privileged mode**
Instructions tsop 1 and tsop 2 can be the same **instruction** as embodied herein as or 1,1,1 but they can also be separate instructions. These instructions interact with bits 19 and 21 of the...

...thread
switch control register 320 is zero, the priority for both threads is set to med um and the effect of the or xx.,x **instructions** on the priority is **blocked**. If an external interrupt request is active, and if the corresponding thread's priority is low, that thread's priority is set to medium. The...

18/3,K/36 (Item 36 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00488444 **Image available**
METHOD AND APPARATUS FOR ACCESSING AND EXECUTING THE CONTENTS OF PHYSICAL MEMORY FROM A VIRTUAL MEMORY SUBSYSTEM
Patent Applicant/Assignee:
PHOENIX TECHNOLOGIES LIMITED,
ZILMER Matthew E,
PHAN Quang,
Inventor(s):
ZILMER Matthew E,
PHAN Quang,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9919796 A1 **19990422**
Application: WO 98US21228 19981009 (PCT/WO US9821228)
Priority Application: US 97947990 19971009
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GD GE GH GM HR HU ID IL IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG US UZ VN YU ZW GH GM KE LS MW SD SZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK

ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG
Publication Language: English
Fulltext Word Count: 21329

Patent and Priority Information (Country, Number, Date):
Patent: ...**19990422**
Fulltext Availability:
Detailed Description
Publication Year: **1999**

Detailed Description
... are protected from it. This ensures that user mode services and applications will not write over each other's memory, or execute each other's **instructions**. **Kernel mode** services and applications are protected in a similar way. If an attempt to access memory outside of a program@s allocated virtual space occurs, the...of predetermined instruction sequences, transfer control to the one of the plurality of predetermined instruction sequences, and process the one of the plurality of predetermined **instruction** sequences from the virtuai memory.

BRIEF DESCRIPTION OF THE DRAWINGS
Figure I is a system **block** diagram of an exemplary processor system in which the apparatus and method of the present invention is used.

Figure 2 is an overall functional block...aware applications.

Terms
APM B10S
System BIOS which provides power management functions adhering to the APIA specification (currently at revisionlevel 1.2)
Kernel Made
A **privileged** processor **mode** in which the NT system code nins.
A **kernel mode thread** has access to all Ito and system memory
Power Management Kernel Driver (PM Driver)
Provides access to the BIOS ROM, BIOS Data Area and the...

18/3,K/37 (Item 37 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00483529
CRYPTOGRAPHIC CO-PROCESSOR
COPROCESSEUR CRYPTOGRAPHIQUE

Patent Applicant/Assignee:
INFORMATION RESOURCE ENGINEERING INC,
KAPLAN Michael M,
DOUD Robert Walker,
KAVSAN Bronislav,
OBER Timothy,
REED Peter,
Inventor(s):
KAPLAN Michael M,
DOUD Robert Walker,
KAVSAN Bronislav,
OBER Timothy,
REED Peter,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9914881 A2 **19990325**
Application: WO 98US19316 19980916 (PCT/WO US9819316)
Priority Application: US 9759082 19970916; US 9759839 19970916; US 9759840 19970916; US 9759841 19970916; US 9759842 19970916; US 9759843 19970916; US 9759844 19970916; US 9759845 19970916; US 9759846 19970916 ; US 9759847 19970916
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GE GH GM HR HU ID IL IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT UA UG US UZ VN YU ZW GH GM KE LS MW SD SZ UG ZW AM AZ BY KG KZ MD RU TJ TM AT BE CH CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN GW ML MR NE SN TD TG
Publication Language: English
Fulltext Word Count: 95649

Patent and Priority Information (Country, Number, Date):
Patent: ...19990325
Fulltext Availability:
Detailed Description
Publication Year: 1999

Detailed Description
... hardware. In most systems, security is achieved by software, such as described above, which is not entirely secure because there is no security hardware to **block** access to the security software by an intruder. Another problem associated with software encryption algorithms is that some of the software encryption algorithms run slower...Note that any DMA transfer which is already running will never be preempted. The only way for a DMA transfer to be aborted in mid-**stream** is for a Host bus error to occur (e.g. a PCI abort due to a Parity error) or for the DSP to force an...1 5: 0]
PCI Local Address High Register (PCB31AH)

If the transfer is between a PCI Host and External Memory (case 3), then this register **holds** the IO most-significant bits of the External Address [25:16] and the most-significant bit will be set to 'I'. If the transfer is...loaded as part of the crypto-context.

Hash Paddin
Controls are also provided to determine what is done at the end of the input data **stream**. If the operation is to be resumed at a later time, then the operation should be defined to exclude 'final' processing. In this case, no...

...512-bit padding hash block will be added to contain the length. Any additional bytes required to fill out the last or extra 512-bit **blocks** are set to zero.

As previously described, the same input data **stream** is internally buffered by both the hash and encryption sections, depending on the data flow of the operation selected. In the case of hash-encrypt...

...negotiated for an IP connection. The pre-computed inner hash is loaded as the initial state of the hash algorithm before processing the input data **stream**. At the end of the
56
input data, normal 'final' processing is done, then the resulting digest is used as data for an additional round...sequence of steps is simplified if operations are performed serially with no overlap.

Final processing for hashing operations, and padding of a short trailing crypto **block**, is initiated automatically. The end of the input data **stream** for an operation is determined by counting the number of bytes input and comparing to the byte length entered as part of the operation control...Address (READ ONLY)
MCIA ... ::.

Oxl8AO Not Visible Not Visible
151413121110987654321O
0
1L0
1 = DSP low Byte Enable
LI = DSP high Byte Enable
Reserved
Reset Reason/**Instruction** Register (RSTREASON)
This 16-bit Read/Write register, as shown in the table below, allows the DSP processor to determine the offending address and memory...

...section previous described on Kemal Mode Control for more information about the protection features of the CryptlC.

The four memory types are as follows.

PM **Instruction** Fetch An **instruction** fetch was made either to
one of the four **blocks** of
Kernel ROM (i.e. the 4 lsb's of PMOVLAY were set to OxC, D, E or
F and a fetch was made between...O O I O I O I O 70
14-bit Memory Address which caused violation
I 0 Memory Type which caused violation
00 = PM **Instruction** Fetch
0 1 = DM Data Fetch
10 = PM Data Fetch
1 1 = Protected Registers@ernel RAM data fetch
External Memory Configuration Register (EXMEMCFG)
This 16...1.

Host wrote Command The Host Processor has just written the
least-significant word of the Application Registers mailbox. This
notifies the DSP of this **event**.

DSP wrote Command The D SP has just written the least-significant word of
the Application Registers mailbox. This notifies the Host of this event
...has been Queued
I = Master DMA Memory Transfer is Done
I = Host Wrote Command Register
I = Hash/Encrypt Context 0 is Done
I = Hash/Encrypt **Context** I is Done
1 = Host External Memory Conflict
I = Host Issued Interrupt
I = HasivEncrypt Error
Reserved
DSP Masked Status Register (DMSTAT)
This 16-bit Read...the CGX Kernel context and invokes the RTI operation.
The Status registers and Program Counter are automatically popped off the
hardware stack at the RTI **instruction**. The return-frominterrupt
operation vectors back into the CGX **Kernel mode**, at which
point the Kernel Protection logic will resume and the interrupted CGX
command processing will continue.

104
Storage Management
The CGX Command Processor is...by the application, the CGX Kernel is
entered by setting the PMOVLAY register to OxOOOF and then executing a
CALL Ox2OOO instruction. From the call **instruction** and some
associated hardware logic and signals, the internally ROM encoded CGX
Kernel program **block** is overlaid in the upper portion of the DSP's
program space. This allows the CGX Kernel code to become active and
transparent to the...is a specific call to address Ox2OOO instruction
with the PMOVLAY register set to OxOOOF. The AR register must contain a
pointer to the kernel **block** which defines the command requested.

When the software call **instruction** occurs, the special hardware logic (along with some hardware signals and counters) overlay the ROM encoded CGX Kernel
157
program block over a portion of...block (described below).

The data buffers, datain (plain-text) and dataout (cipher-text), can share the same address. Moreover, the chip currently only supports cipher-**block** symmetrical algorithms but the encrypt interface provides
for the addition of a **stream** interface. It does this by defining datain
and dataout as unsigned. char pointers and the data length of datain to be in bytes. Allowing the byte count makes the interface portable for the later addition of **stream** based algorithms. However, if the algorithm to be used in the encrypt operation is cipher-**block** based the
byte count must be evenly divisible by 8, any fragments are ianored by the operation.

206
Prior to invoking the encryption command the...block (described below).
The data buffers, datain (plain-text) and dataout (cipher-text), can share the same address. Moreover, the chip currently only supports cipher-**block** symmetrical algorithms but the decrypt interface provides
for the addition of a **stream** interface. It does this by defining datain
and dataout as ...char pointers and the data length of datain to be in bytes. Mowing the byte count makes the interface portable for the later addition of **stream** based algorithms. However, if the algorithm to be used in the decrypt operation is cipher-**block** based the
byte count must be evenly divisible by 8, any fragments are ignored by the operation.

Prior to invoking the Decrypt command the application...

18/3,K/38 (Item 38 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00418748 **Image available**
SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

. Patent Applicant/Assignee:
INTERTRUST TECHNOLOGIES CORP,
Inventor(s):
GINTER Karl L,
SHEAR Victor H,
SIBERT W Olin,
SPAHN Francis J,
VAN WIE David M,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9809209 A1 **19980305**
Application: WO 97US15243 19970829 (PCT/WO US9715243)
Priority Application: US 96706206 19960830
Publication Language: English
Fulltext Word Count: 195626

Patent and Priority Information (Country, Number, Date):
Patent: ...19980305
Fulltext Availability:
Detailed Description
Publication Year: 1998

Detailed Description
... rights of citizens and
organizations who use this information (also 'electronic' or
"digital") highway.

2
Electronic Content
Today, virtually anything that can be represented by
instructions can be formatted into electronic digital information.
Television, cable, satellite transmissions, and on-line services
transmitted over telephone lines, compete to distribute digital
information and...other
parties. Distribution may be by, for example, physical media
delivery, broadcast and/or telecommunication means, and in the
form of 'static" files and/or **streams** of data. VDE mav also be
used, for example, for multi-site "real-time' 'interaction such as
teleconferencing, interactive games, or on-line bulletin boards...be
protected.

- 176
Rules and Contents' Can Be Separately Delivered
As mentioned above, virtual distribution environment 100
"associates" content with corresponding "rules and controls," and
prevents the content from being used or accessed unless a set of
corresponding 'rules and controls' is available. The distributor
106 doesn't need to deliver...as a "computer prograni(s)" "embedded'
within
- 184 chip 504. Firmware 508 makes the hardware 506 work.

Hardware 506 preferably contains a processor to perform **instructions** specified by firmware 508. "Hardware" 506 also contains long-term and short-term memories to store information securely so it can't be tampered with...control security-relevant aspects of other components in the CPU/SPU. The "SPU" mode in this
- 224
to be fetched from secure memory 532, 534 -- **preventing** execution based on "mixed" secure and non-secure **instructions**.

In the "SPU" mode, mode interface switch 2658 may, in one example embodiment, disconnect or otherwise block external accesses ca=.ed over bus 652 from...

...or otherwise
block access by microprocessor 2652 to some external memory and/or other functions carried over bus 652. Mode 'interface
'tch 2658m' this example **prevents** other CPU
swi
operations/**instructions** from exposing the contents of secure memory 532, 534.

In the example shown in Figure 9A, the mode control of mode interface switch 2658 is...exposed by microprocessor 2652 operations that occur in the
"normal" mode. This may be accomplished either by hardware mechanisms that protect against such exposure, software **instructions** executed in "SPU" mode that clear, reinitialize, and otherwise reset during such transitions, or a combination of both.

In some example inlDlementations, 'interrupts may be...maintain security in such multi-package versions, it
may be necessary to enclose all the packages and their interconnections in an external physical tarnper-resistant **barrier**.

- 230
Initialization of Integrated CPU/SPU
Instructions and/or data may need to be loaded into CPU/SPU 2650 before it can operate effectively as an SPU 500.

This may occur during...precisely as does microprocessor 2652 with respect to power-on reset and other external conditions. Under such conditions, only execution of the "enable SPU mode" **instruction** or otherwise requesting SPU mode under program control would cause "SPU" mode to be entered. Additionally, a mechanism could be provided to permit microprocessor 2652...

...its operation.

In the preferred embodiment, CPU/SPU 2650 would, when SPU mode has not yet been established, begin operating in SPU mode by fetching **instructions** from secure non-volatile memory 532, thereby ensuring a consistent initialization sequence and **preventing** SPU dependence on any information held outside CPU/SPU 2650. This approach permits secret initialization information (e.a., keys for validating digital signatures on additional...microprocessor 2652 indicatina entrv to, exit from, and control of SPU mode.

Switch 2663 'in this example reco a specific
gm
indication (e.g., an **instruction** fetch access to a designated address in the secure memory 532) as the equivalent to the enable 'SPU mode" **instruction**. Upon recognizing such an indication, it may isolate the CPU/SPU 2650 from external buses and interfaces 2664, 2665, and 2667 such that any external...
...such as DMA cycles, would be "held" until the switch
235
single access to a specific location in secure memory 532 to complete.

The single **instruction** fetched from the designated location performs a control operation (a cache flush, for example), that can only be performed in microprocessor 2652's most **privileged** operating **mode**, and that has an effect visible to switch 2663.

Switch 2663 awaits the occurrence of this event, and if it does not occur within the expected number of cycles, does not enter "SPU" mode.

Occurrence of the control operation demonstrates that microprocessor 2652 is executing in its most **privileged** "normal" mode and therefore can be trusted to execute successfully the #I enter'SPU mode" sequence of **instructions** stored in secure memory 532. If microprocessor 2652 were not executing in its most **privileged** **mode**, there would be no assurance that those **instructions** would execute successfully. Because switch 2663 isolates microprocessor 2652 from external signals (e.g., interrupts) until "SPU" mode is successfully initialized, the entry instructions can be guaranteed to complete successfully.

236
Following the initial **instruction**, switch 2663 can enter of partial SPU mode," in which a restricted area of ROM 532 and RAM 534 may be accessible. Subsequent **instructions** in secure memory 532 may then be executed by microprocessor 2652 to

place it into a known state such that it can perform SPU functions -- saving any previous state in the restricted area of RAM 534 that is accessible. After the known state is established, an **instruction** may be executed to deliver a further indication (e.g., a reference to another designated memory location) to switch 2663, which would enter "SPU" mode...

...mode. Once in "SPU" mode, switch 2663 permits access to all of ROM 532, RA"M 534, and other devices in SPU chip 2660.

The **instructions** executed during "partial SPU" mode must be carefully selected to ensure that no similar combination of **instructions** and processor state could result 'in a control transfer
out of the protected SPU code in ROM 532 or RAMT 534. For example, internal debugging...

...disabled or reinitialized to ensure that previously created MMU data
- 237
structures would not permit SPU memory accesses to be compromised. The requirement that the **instructions** for "partial SPU mode" run in the microprocessor 2652's most **privileged mode** is necessary to ensure that all its processor control functions can be effectively disabled.

The switch 2663 provides additional protection against tampering by ensuring that control functions) may be **prevented** by switch 2663 unless they had been explicitly enabled by **instructions** executed after "SPU" mode is entered.

. 238
To leave SPU mode and return to normal operation, the instructions executing in "SPU" mode may provide a...
...such an embodiment, switch 2663 may not implement partial SPU mode, but may instead enter SPU mode directly and ensure that the address from which **instructions** would be fetched by microprocessor 2652 (specific to microprocessor 2652's architecture) results in accesses to appropriate locations in the SPU memory 532. This could...users.

Computer systems are usually made up of several different hardware components. These hardware components include, for example.

a central processing unit (CPU) for executing **instructions**;
- 246
an array of main memory cells (e.g., 'RAIM" or 'ROM") for

storing **instructions** for execution and data acted upon or parameterizing those **instructions**; and
one or more secondary storage devices (e.g., hard disk drive, floppy disk drive, CD-ROM drive, tape reader, card reader, or "flash' memory...discover details of processes; using self-generating" code (based on the output of a co-sine transform, for example) such that detailed and/or complete **instruction** sequences are not stored explicitly on storage devices and/or in active memory but rather are generated as needed; using code that "shuffles' memory locations...isuq 'uLioj.iad oz 2uuudaid -io/pu-e 12umuopad u-i asnioj sdriqsuoT -elei DPZST/L6SfI/1Jd 6OZ60/86 OM
information that identifies associated basic **instructions** and said intrinsic data for access, correlation and/or validation purposes;
0 required and/or optional parameters for use with basic **instructions** and said intrinsic data;
0 information defining relationships to other methods;
0 data elements that may comprise data values, fields of information, and/or the...

18/3,K/39 (Item 39 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00393569 **Image available**
DISTRIBUTED STATUS SIGNALING SYSTEM FOR MULTI-THREADED DATA STREAM TRANSPORT CONTROL

Patent Applicant/Assignee:
DIAMOND MULTIMEDIA SYSTEMS INC,
Inventor(s):
LEWIS Adrian,
GIFFORD James K,
BEGUR Sridhar,
SPENCER Donald J,
KILBOURN Thomas E,
GOCHNAUER Daniel B,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9734312 A1 **19970918**
Application: WO 97US3345 19970306 (PCT/WO US9703345)
Priority Application: US 96615682 19960313
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CN JP AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Fulltext Word Count: 23624

Patent and Priority Information (Country, Number, Date):
Patent: ...19970918
Fulltext Availability:
Detailed Description
Publication Year: 1997

Detailed Description
... streams enables the effectively concurrent parallel transfers of digital data streams to and through the channel controller. This substantially alleviates the need for host application **context** switches and most host user to interrupt **kernel mode** switches while providing real-time transport support for multiple concurrent data **streams**

A yet further advantage of the present invention is that it provides dynamic support of variable rate real time signal processing by ensuring effectively continuous...concurrent data and control signal transfers between the PCI bus 20 and system 22. These concurrent transfers are preferably managed as independent pairs of data **streams** and control threads of execution where each **stream** is composed of discrete **block** transfers, consisting of one or more bytes of data, through the I/0 channel controller 26. Data **blocks** of concurrently supported **streams** are interleaved among one another to effectively establish concurrent **stream** data transport

In accordance with the present invention, the interleave of data **blocks** is determined by ...core 26 reflects the respective demand transport rates of the individual data streams as determined from the nature of the stream data transported. Consequently, where **stream** data represents a constant data frequency audio signal **stream**, the interleave of data **blocks** in the audio data **stream** may vary, though generally maintaining a fixed through-put rate reflective of the data stream transfer rate independently demanded by an audio stream coder/decoder...transfer

The host processor establishes a control thread in the main memory space 54 by constructing one or more linked bus transfer unit (BTU) control **blocks** to associate the **stream** data as provided in the main memory space 54 with the control thread. The host processor 12, in execution of the I/O channel controller...stream 72 may be subsequently transferred again by the digital signal processor to any directly connected serial bus peripheral 40, 42

Bus transfer unit control **blocks** that define a
control thread for transferring the DSP RAM stored data
stream 69, 80 can ...36
to operate by definition independent of one another with
respect to the use of the DSP RAM 34

The DSP bus transfer unit control **blocks** are
defined to control the transfer of one or more data
streams through and under the control of the I/ ...116 to the
memory 122 or external
hardware 124 of the controller peripheral layer 118.
The device driver 112 establishes one or more BTU
control **blocks** within the memory 106 defining a control
thread necessary to implement the data **stream** transfer.
The initial BTU of the control thread is programmed by
the device driver 112 into the I/0 channel controller
and a signal is...channel controller core
116. Preferably, the message is acknowledged by the DSP
120, the requested function is initialized, a portion of
the externally provided data **stream** is received and
processed by the DSP 120 and stored in a known address
block within the memory 122. An enable signal 136 ...DSP side BTU
is loaded into the I/O
channel controller core 116 and left disabled.
Meanwhile, a next portion of the externally provided
data **stream** is received and processed into another known
address **block** within the memory 122 that is referenced
by the next DSP side BTU. Again, once this address
block has been filled with processed ...136. In general, the two address
blocks in
the memory 122 are successively alternatingly used in
successive alternation for the processing of a single
data **stream**

When the address **block** specified by the current
host side BTU in the memory 106 is filled by the
processed incoming data stream, the BTU is complete and
the...

18/3,K/40 (Item 40 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00393493 **Image available**
MULTIPLE PARALLEL DIGITAL DATA STREAM CHANNEL CONTROLLER ARCHITECTURE
Patent Applicant/Assignee:
DIAMOND MULTIMEDIA SYSTEMS INC,

Inventor(s):
GIFFORD James K,
BEGUR Sridhar,
LEWIS Adrian,
SPENCER Donald J,
KILBOURN Thomas E,
GOCHNAUER Daniel B,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9734236 A1 **19970918**
Application: WO 97US3665 19970306 (PCT/WO US9703665)
Priority Application: US 96614729 19960313
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CN JP AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 23993

Patent and Priority Information (Country, Number, Date):
Patent: ...19970918
Fulltext Availability:
Detailed Description
Publication Year: 1997

Detailed Description
... streams enables
the effectively concurrent parallel transfers of digital data streams to and through the channel controller.

This substantially alleviates the need for host application **context** switches and most host user to interrupt **kernel** **mode** switches while providing real-time transport support for multiple concurrent data **streams**.

is A yet further advantage of the present invention is that it provides dynamic support of variable rate real time signal processing by ensuring effectively...concurrent data and control signal transfers between the PCI bus 20 and system 22. These concurrent transfers are preferably managed as independent pairs of data **streams** and control threads of execution where each is **stream** is composed of discrete **block** transfers, consisting of one or more bytes of data, through the 1/0 channel controller 26. Data **blocks** of concurrently supported **streams** are interleaved among one another to effectively establish concurrent **stream** data transport.

In accordance with the present invention, the interleave of data **blocks** is determined by the

controller sub-system 22 and, in general, independent of the execution of the host processor 12, The interleave of data blocks...

...core 26 reflects the respective demand transport rates of the individual data streams as determined from the nature of the stream data transported. Consequently, where **stream** data represents a constant data frequency audio signal **stream**, the interleave of data **blocks** in the audio data **stream** may vary, though generally maintaining a fixed through-put rate reflective of the data stream transfer rate independently demanded by an audio stream coder/decoder...transfer.

The host processor establishes a control thread in the main memory space 54 by constructing one or more linked bus transfer unit (BTU) control **blocks** to associate the **stream** data as provided in the main memory space 54 with the control thread. The host processor 12, in execution of the I/O channel controller...stream 72 may be subsequently transferred again by the digital signal processor to any directly connected serial bus peripheral 40, 42.

Bus transfer unit control **blocks** that define a control thread for transferring the DSP RAM stored data **stream** 69, 80 can be provided in either system main or DSP memory 14, 34. Preferably, the DSP control thread bus transfer unit control blocks are...

...36
to operate by definition independent of one another with respect to the use of the DSP RAM 34.

The DSP bus transfer unit control **blocks** are defined to control the transfer of one or more data **streams** through and under the control of the I/O channel controller core 62 to any of the external hardware interfaces 74 of the serial and...116 to the memory 122 or external hardware 124 of the controller peripheral layer 118.

The device driver 112 establishes one or more BTU control **blocks** within the memory 106 defining a control thread necessary to implement the data **stream** transfer.

The initial BTU of the control thread is programmed by the device driver 112 into the I/O channel controller 116 and a signal...channel controller core 116. Preferably, the message is acknowledged by the DSP

120, the requested function is initialized, a portion of the externally provided data **stream** is received and processed by the DSP 120 and stored in a known address **block** within the memory 122. An enable signal 136 is then provided to the I/O channel controller core 116 to enable the initial DSP side...

...DSP side BTU is loaded into the I/O channel controller core 116 and left disabled.

Meanwhile, a next portion of the externally provided data **stream** is received and processed into another known address **block** within the memory 122 that is referenced by the next DSP side BTU. Again, once this address block has been filled ...136. in general, the two address blocks in the memory 122 are successively alternatingly used in successive alternation for the processing of a single data **stream**.

When the address **block** specified by the current host side BTU in the memory 106 is filled by the processed incoming data stream, the BTU is complete and the...

18/3,K/41 (Item 41 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00393492 **Image available**
MULTI-THREADED FIFO POOL BUFFER AND BUS TRANSFER CONTROL SYSTEM
Patent Applicant/Assignee:
DIAMOND MULTIMEDIA SYSTEMS INC,
Inventor(s):
BEGUR Sridhar,
GIFFORD James K,
LEWIS Adrian,
SPENCER Donald J,
KILBOURN Thomas E,
GOCHNAUER Donald B,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9734235 A1 **19970918**
Application: WO 97US3667 19970306 (PCT/WO US9703667)
Priority Application: US 96614659 19960313
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CN JP AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English

Fulltext Word Count: 23960

Patent and Priority Information (Country, Number, Date):
Patent: ...19970918
Fulltext Availability:
Detailed Description
Publication Year: 1997

Detailed Description
... for repeatedly
evaluating the status information and providing for the prioritied selection of a first data transfer device and a predetermined one of the memory **blocks**.

A computer system utilizing the multiple parallel digital data **stream** channel controller of the present invention can thus support the concurrent real-time transfer of a plurality of I/O data streams to and from...

...streams enables
the effectively concurrent parallel transfers of digital data streams to and through the channel controller. This substantially alleviates the need for host application **context** switches and most host user to interrupt **kernel** **mode** switches while providing real-time transport support for multiple concurrent data **streams**, A yet further advantage of the present invention is that it provides dynamic support of variable rate real time signal processing by ensuring effectively continuous...concurrent data and control signal transfers between the PCI bus 20 and system 22. These concurrent transfers are preferably managed as independent pairs of data **streams** and control threads of execution where each **stream** is composed of discrete **block** transfers, consisting of one or more bytes of data, through the I/O channel controller 26. Data **blocks** of concurrently supported **streams** are interleaved among one another to effectively establish concurrent **stream** data transport.

In accordance with the present invention, the interleave of data **blocks** is determined by the controller sub-system 22 and, in general, independent of the execution of the host processor 12. The interleave of data blocks...
...core 26 reflects the respective demand transport rates of the individual data streams as determined from the nature of the stream data transported. Consequently, where **stream** data represents a constant data frequency audio signal **stream**, the interleave of data **blocks** in

the audio data **stream** may vary, though generally maintaining a fixed through-put rate reflective of the data stream transfer rate independently demanded by an audio stream coder/decoder...transfer.

The host processor establishes a control thread in the main memory space 54 by constructing one or more linked bus transfer unit (BTU) control **blocks** to associate the **stream** data as provided in the main memory space 54 with ...stream 72 may be subsequently transferred again by the digital signal processor to any directly connected serial bus peripheral 40, 42, Bus transfer unit control **blocks** that define a control thread for transferring the DSP RAM stored data **stream** 69, 80 can be provided in either system main or DSP memory 14, 34. Preferably, the DSP control thread bus transfer unit control blocks are...

...36

to operate by definition independent of one another with respect to the use of the DSP RAM 34.

The DSP bus transfer unit control **blocks** are defined to control the transfer of one or more data **streams** through and under the control of the I/O channel controller core 62 to any of the external hardware interfaces 74 of the serial and...116 to the memory 122 or external hardware 124 of the controller peripheral layer 118.

The device driver 112 establishes one or more BTU control **blocks** within the memory 106 defining a control thread necessary to implement the data **stream** transfer.

The initial BTU of the control thread is programmed by the device driver 112 into the I/O channel controller 116 and a signal...channel controller core 116, Preferably, the message is acknowledged by the DSP 120, the requested function is initialized, a portion of the externally provided data **stream** is received and processed by the DSP 120 and stored in a known address **block** within the memory 122, An enable signal 136 is then provided to the I/O channel controller core 116 to enable the initial DSP side **stream** is received and processed into another known address **block** within the memory 122 that is referenced by the next DSP side BTU. Again, once this address block has been filled with processed data, the...

...136, In general, the two address blocks in the memory 122 are successively alternatingly used in

successive alternation for the processing of a single data **stream**.

- 29
When the address **block** specified by the current host side BTU in the memory 106 is filled by the processed incoming data stream, the BTU is complete and the...

18/3,K/42 (Item 42 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00393491 **Image available**
METHOD AND APPARATUS SUPPORTING DEMAND DRIVEN MULTIPLE PARALLEL DIGITAL DATA STREAM TRANSPORT
Patent Applicant/Assignee:
DIAMOND MULTIMEDIA SYSTEMS INC,
Inventor(s):
SPENCER Donald J,
GIFFORD James K,
BEGUR Sridhar,
LEWIS Adrian,
KILBOURN Thomas E,
GOCHNAUER Daniel B,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9734234 A1 **19970918**
Application: WO 97US3666 19970306 (PCT/WO US9703666)
Priority Application: US 96596921 19960313
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
CN JP AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE
Publication Language: English
Fulltext Word Count: 23913

Patent and Priority Information (Country, Number, Date):
Patent: ...19970918
Fulltext Availability:
Detailed Description
Claims
Publication Year: 1997

Detailed Description
... streams enables
the effectively concurrent parallel transfers of digital data streams to and through the channel controller.

This substantially alleviates the need for host application **context** switches and most host user to interrupt **kernel mode** switches while providing real-time transport support for multiple concurrent data **streams**.

A yet further advantage of the present invention is that it provides dynamic support of variable rate real time signal processing by ensuring effectively continuous...concurrent data and control signal transfers between the PCI bus 20 and system 22. These concurrent transfers are preferably managed as independent pairs of data **streams** and control threads of execution where each **stream** is composed of discrete **block** transfers, consisting of one or more bytes of data, through the I/O channel controller 26. Data **blocks** of concurrently supported **streams** are interleaved among one another to effectively establish concurrent **stream** data transport.

In accordance with the present invention, the interleave of data **blocks** is determined by the
- 14
controller sub-system 22 and, in general, independent of the execution of the host processor 12. The interleave of data...

...26 reflects the respective demand transport rates of the individual data streams as determined f rom the nature of the stream data transported. Consequently, where **stream** data represents a constant data frequency audio signal **stream**, the interleave of data **blocks** in the audio data **stream** may vary, though generally maintaining a fixed through-put rate reflective of the data stream transfer rate independently demanded by an audio stream coder/decoder...transfer.

The host processor establishes a control thread in the main memory space 54 by constructing one or more linked bus transfer unit (BTU) control **blocks** to associate the **stream** data as provided in the main memory space 54 with the control thread. The host processor 12, in execution of the I/O channel controller...

...stream 72 may be subsequently transferred again by the digital signal processor to any directly connected serial bus peripheral 40, 42.

Bus transfer unit control **blocks** that define a control thread for transferring the DSP RAM stored data **stream** 69, 80 can be provided in either system main or

DSP memory 14, 34. Preferably, the DSP control thread bus transfer unit control blocks are...

...36
to operate by definition independent of one another with respect to the use of the DSP RAM 34.

The DSP bus transfer unit control **blocks** are defined to control the transfer of one or more data **streams** through and under the control of the I/O channel -22
controller core 62 to any of the external hardware interfaces 74 of the serial...116 to the memory 122 or external hardware 124 of the controller peripheral layer 118.

The device driver 112 establishes one or more BTU control **blocks** within the memory 106 defining a control 3b thread necessary to implement the data **stream** transfer.

The initial BTU of the control thread is programmed by the device driver 112 into the I/O channel controller 116 and a signal...channel controller core 116. Preferably, the message is acknowledged by the DSP 120, the requested function is initialized, a portion of the externally provided data **stream** is received and processed by the DSP 120 and stored in a known address **block** within the memory 122. An enable signal 136 is then provided to the I/ ...DSP side BTU is loaded into the I/O channel controller core 116 and left disabled.

Meanwhile, a next portion of the externally provided data **stream** is received and processed into another known address **block** within the memory 122 that is referenced by the next DSP side BTU. Again, once this address block has been filled with processed data, the...

...136. In general, the two address blocks in the memory 122 are successively alternatingly used in successive alternation for the processing of a single data **stream**.

When the address **block** specified by the current host side BTU in the memory 106 is filled by the processed incoming data stream, the BTU is complete and the...

Claim
... second interfaces;
c) a buffer memory coupled between said first

buffer and each of said second buffers, said buffer memory including a plurality of memory **blocks** permitting storage of third variable segment portions of the multiple data **streams**; and
d) a **stream** controller coupled to said first interface, to each of said second interfaces, and to said buffer memory, said controller being responsive to the respective data lengths of said second variable segment portions, said **stream** controller selectively enabling transfers of data between a predetermined memory **block** of said buffer memory and a predetermined second buffer.

6 The data transfer controller of claim 5 wherein said stream controller is responsive to the respective data lengths of said third variable segment portions, said **stream** controller selectively enabling transfers of data between said first buffer and said plurality of memory **blocks** so as to support the availability of data in said predetermined memory block subject to the transfers of data with respect to said predetermined 2.5 second buffer.

7 The data transfer controller of Claim 6 wherein said **stream** controller supports the availability of data in each of said memory **blocks** subject to the combined transfers of data with respect to said second buffers.
B. The data transfer controller of Claim 7 wherein said stream controller...

...with the respective data transfers of said second variable segment portions through said second buffers.

9 The data transfer controller of claim 8 wherein said **stream** controller operates sets of one or more of said memory **blocks** as independent circular FIFO buffers.

10 The data transfer controller of Claim 9 wherein said first and second buffers are FIFO buffers.
11 A computer...

18/3,K/43 (Item 43 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00387760 **Image available**
METHOD AND SYSTEM FOR IMPLEMENTING DATA TRANSFERS
PROCEDE ET SYSTEME DE REALISATION DE TRANSFERTS DE DONNEES

Patent Applicant/Assignee:
THE TRUSTEES OF PRINCETON UNIVERSITY,
Inventor(s):
BLUMRICH Matthias A,
FELTEN Edward W,
LI Kai,
DUBNICKI Cezary,
Patent and Priority Information (Country, Number, Date):
Patent: WO 9728503 A1 **19970807**
Application: WO 97US1108 19970127 (PCT/WO US9701108)
Priority Application: US 96595792 19960202
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AL AM AT AU AZ BA BB BG BR BY CA CH CN CU CZ DE DK EE ES FI GB GE HU IL IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK TJ TM TR TT UA UG UZ VN KE LS MW SD SZ UG AM AZ BY KG KZ MD RU TJ TM AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE BF BJ CF CG CI CM GA GN ML MR NE SN TD TG
Publication Language: English
Fulltext Word Count: 14454

Patent and Priority Information (Country, Number, Date):
Patent: ...19970807
Fulltext Availability:
Detailed Description
Publication Year: 1997

Detailed Description
... Prevents moving a page of data out of memory by virtual memory processing.

System Call - A function call to the operating system.

This causes a **context** switch to the operating system, System Level - The operating system. System level code normally uses the **privileged mode** of the CPU operation which allows the full **instruction** set of the processor to be used.

a User-level - Applications. User-level code normally does not use the **privileged mode** of the CPU operation, so a restricted set of **instructions** is allowed.

DMA is a common technique for routing data directly between memory and an I/O device without requiring intervention by the CPU to...atomicity of DMA transfer initiations, to create virtual memory mappings, and to maintain memory proxy mappings during virtual memory paging.

The first invariant, Il must **hold** regardless of source and destination devices.

11: If a LOAD **instruction** initiates a UDMA data transfer, then the destination address and the byte count must have been STORED by the same process,
The next three invariants...

18/3,K/44 (Item 44 from file: 349)
DIALOG(R)File 349:PCT FULLTEXT

00149051
MULTIPROCESSING METHOD AND ARRANGEMENT
PROCEDE ET AGENCEMENT A MULTIPROCESSEUR
Patent Applicant/Assignee:
AMERICAN TELEPHONE & TELEGRAPH COMPANY,
Inventor(s):
SCHAN Edward Paul Jr,
STRELIOFF Brian Kent,
Patent and Priority Information (Country, Number, Date):
Patent: WO 8805944 A1 **19880811**
Application: WO 87US1829 19870727 (PCT/WO US8701829)
Priority Application: US 8789 19870206
Designated States:
(Protection type is "patent" unless otherwise stated - for applications prior to 2004)
AT BE CH DE FR GB IT JP LU NL SE
Publication Language: English
Fulltext Word Count: 9571

Patent and Priority Information (Country, Number, Date):
Patent: ...**19880811**
Fulltext Availability:
Detailed Description
Publication Year: **1988**

Detailed Description
... the process for execution to the master processor. Execution of the transferred process resumes on the master processor substantially at a point at which execution **stopped** on the slave processor., Illustrativelyt execution of an interrupted **instruction** of the process is either restarted or merely completed on master processor. In the former cases the attempt to enter privileged mode is repeated on...the invention.

Typicallyr the 3B15 system uses two modesr or

levelsr of operation of the WE 32100 microprocessor: a user mode for executing user program **instructionsr** and a **privileged mode** for executing functions such as operating system **instructions** that have the potential for corrupting shared system resources such as hardware resources. There are two mechanisms for entering **privileged mode** from user mode: a process switch mechanism and a **system call** mechanism. The **system call** mechanism is also known by names such as a supervisory call and an ...a means of controlled entry into a
35 function by installing on a processor a new processor status word (PSW) and program counter (PC). The **system call** mechanism is used by explicit operating **system** GATE **calls** to transfer control from user to **privileged mode**, and is; also used to handle 'normal' system exceptions.

('Quick' interrupts, which would also be handled by this mechanism, are not used by the UNIX...

...exception is an error condition--a fault or a trap--other than an interrupt. Normal exceptions constitute most system exceptions. The normal exception handlers are **privileged-mode** functions.

The **system call** mechanism uses the execution stack of the present process; that is, a normal exception handler or a function called via a GATE instruction uses for...
...privileged state will not become corrupted. Therefore, prior to making a legal entry of a privileged function, i.e., before executing a transfer to the **privileged mode** upon the occurrence of a normal exception or a GATE requestr the **system call** mechanism checks the present execution stack pointer value against the execution stack boundary values that are stored in the process control block of the presently...processor 12, or execution of the partially-executed
instruction is merely completed on master processor 12.

Unless the condition that caused the attempt to enter **privileged mode** was a transient faultr execution of that **instruction** on master processor 12 results in the invocation of the operating system service. That service is then provided in a conventional, uniprocessor, manner on master...the physical address of the slave's initial process control block). Also at step 502, a process switch to the slave's initial process control **block** is performed by the conventional PETPS **instruction**.

As part of this process switch, the slave's process control **block** pointer register is set to the physical address of the intermediate portion of the slave's initial process control block, the initial program counter is...

...step 503, which resets the process control block pointer register to the virtual address of the intermediate portion of the slave's initial process control **blockf** and executes the conventional ENBVJMP **instruction** which enables virtual addressing for slave processor 25 and transfers to the virtual address contained in register rO.

The slave virtual-mode startup routine is...This is implemented by calling a slave delete routine (see FIG* 14). at step 1002a Illustratively in this exampler because the address of the faulting **instruction** is already saved-in the process control **block** as result of stack exception handlingr the faulting **instruction** is reexecuted once the process restarts execution on master processor 12. This feature is fairly typical of systems with demand-paged memory management. This results...routine (see FIG. 13) must be invoked to avoid restart problems. This is illustratively accomplished, at step 1001, by removing the address of the faulting **instruction** from the process control **block** and substituting therefor the starting address of the **instruction** restart routiner and storing the removed faulting instruction's address in a variable.

When the **instruction** restart routine completesr it restores the faulting **instruction**'s address in the process control **block**.

Although far less usual than normal exceptions, another exception condition is possible when executing user-mode processes: system error. This category includes things like alignment...

...as the standard, master, stack exception handler deals with MAU restart problems. The instruction restart routine then restores the address of the user process' faulted **instruction** to the process control **block** of the faulted process to cause execution of the faulted **instruction**, at step 1203p and returns at step 1204 to execution of that instruction.
FIG* 14 flowcharts the slave delete routine.

This routine is invoked on...

Data Streaming: Very Low Overhead Communication for Fine-grained Multicomputing

Patrick T. Gaughan
Computer Architecture Research Laboratory
Department of Electrical Engineering
The University of Alabama
Tuscaloosa, AL 35487-0286
E-mail: pgaughan@shadow.carl.ua.edu

Abstract

Recent developments have greatly reduced network latencies in multiprocessor networks. Thus, software overhead is becoming the primary cost of multiprocessor communication. This paper proposes data streaming — a technique which places explicit send and receive instructions in the user code — as a means to cut software overhead to a minimum. Data streaming has the added benefit that it can tighten the coupling between processors by reducing the message size to that of a single data item. This paper presents experimental results that indicate data streaming can cut software overhead to less than one instruction per byte of data transmitted.

1 Introduction

Advances in multiprocessor interconnection networks have greatly reduced the network latency in massively parallel computer architectures[2, 13, 12]. Software overhead has emerged as the primary cost of multiprocessor communication. Traditional message passing uses interrupts and operating system calls to handle communication. This approach is very costly in software overhead due to the required context switches, type matching operations, and buffering.

An alternative approach is to allow the messages themselves to carry pointers[14] (*continuations* [11]) to user specified message handling routines. These messages are called *active messages*[14]. Active messages have the advantage that every message can be serviced immediately and no operating system buffering is necessary. In addition, costly type matching searches are eliminated. However, interrupting the current thread of control requires a context switch and, in some cases, a trap to kernel mode[14]. Such context switches can be costly on modern microprocessors.

The fundamental problem is that message traffic is unpredictable (or assumed to be) in such programming models. The overhead present in software and hardware is the cost of run-time message disambiguation. If we can statically determine message arrival orders and schedule message traffic at compile-time, we can eliminate these costs. In those cases where messages cannot be disambiguated at compile time, we need architectural constructs to allow them to coexist in the network without introducing ambiguities. This paper proposes *data streaming*, a set of architectural and programming techniques that imbed explicit send and receive instructions directly in the user's program. Each send instruction is statically matched to a receive instruction or a set of "equivalent" receive instructions on another node. To allow multiple communication patterns to overlap within the network, distinct virtual networks are supported in hardware that can be mapped to data streams at compile-time.

This paper presents the theoretical foundations and some experimental support for data streaming. Experimental results indicate that data streaming can significantly reduce the costs of communication. Software overhead using data streaming can be reduced to less than one instruction per byte of data transmitted. In addition, for two of the programs examined, software overhead and unhidden network latency were significantly less than the penalty suffered by local cache misses. In all programs examined, the total cost of communication (software overhead and unhidden network latency) was less than 8 cycles per byte transmitted.

2 Stream-based communication

Traditional message passing techniques had large startup and overhead-per-word costs. It was necessary to amortize the startup costs by keeping message sizes large and minimizing the communication between non-local threads. While minimizing communication is still a good idea, sending large blocks of data limits parallelism that can be exploited within a block. For example, during a vector operation, message passing programs typically receive the non-local vector in full, apply the operation, and send the resulting vector to its next destination. A finer grained approach would begin vector operations as the individual vector components arrived. Thus, the operation on the vector can be overlapped with the transmission of the vector from its source. What's more, the resulting vector can begin its journey to its next destination as the results are computed, creating a pipeline of processors all working on different elements of the same

vector. The values are *streamed* into and out of each processing element. There is minimal need for buffering and, in some instances, values are never placed in local memory at all, passing directly from network to register to network.

2.1 Architectural support

Figure 1 shows a diagram of a data streaming multiprocessor. In the diagram, the processor (P) communicates with other processors via a network interface unit (NIU) and a number of routers. The NIU has multiple virtual channels to and from the local router. The NIU is nothing more than a virtual channel multiplexer that controls which virtual channels are visible to the CPU for send and receive instructions. Via NIU coprocessor instructions, the CPU can select input from any of the incoming virtual channels. The channels are independent streams — if one becomes blocked other channels can continue to transmit and receive data. In this way, multiple communication patterns can be overlapped. Stream data is received a word at a time as the data is needed. Every circuit that the processor creates has an associated *stream descriptor* which specifies the virtual network the circuit uses. If the required channel at the destination node is occupied, the routing header will block in the network.

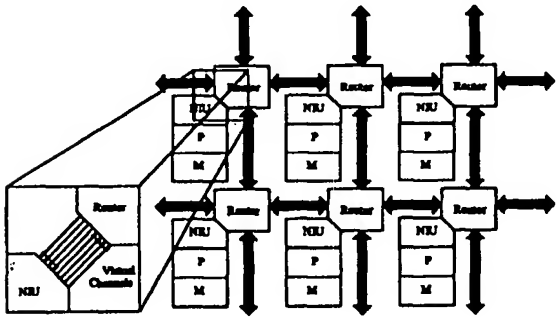

Figure 1: A node in a multiprocessor network

The following instruction set enhancements are proposed to support data streaming:

set_send_sd $rs — Set the stream descriptor of the output stream to the contents of register $rs. If this descriptor is active, the next send will output a data word to the active stream. Otherwise, the next send will specify the destination of the new stream.

send $rs — Send the contents of register $rs to the current output stream. If the required output channel's buffer is full, the instruction stalls the CPU.

sende $rs — Send the contents of register $rs to the current output stream and *end the circuit* (i.e. generate the tail flit). Stall CPU on buffer full.

set_recv_sd $rs — Set the stream descriptor of the input stream to the contents of the $rs register. The next receive instruction will attempt to read data from this virtual channel.

recv $rs — Receive a word of data into $rs from the current input stream. If no data is present, stall until data for that stream arrives.

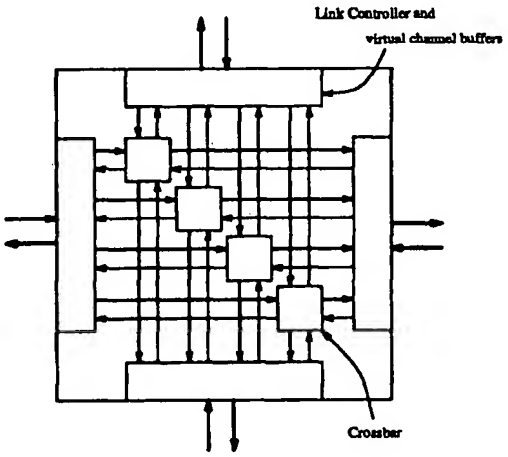

Figure 2: A disjoint stream router

The network routers will implement a disjoint stream network where each virtual channel belongs to a virtual network called a *stream net*. No routing is allowed between these stream nets. Thus, messages with different stream descriptors cannot interfere with each other in the network. The router for a disjoint stream network is much simpler than a more general router for a network with v general purpose virtual lanes on each link. In an n dimensional mesh, a disjoint stream network would require v $n \times n$ crossbars as opposed to a $vn \times vn$ crossbar which is about v times larger.

3 Deadlock-freedom

Data streaming is not a trivial extension of current parallel programming techniques. This section will briefly describe conditions that are sufficient to guarantee deadlock-free, correct operation. A complete development can be found in [7].

Process equivalence: Two circuits are process equivalent if they are processed by a processor in the network in exactly the same way. That is, the processor has no need to distinguish between them, and thus, their arrival order is unimportant. For example, in a global sum reduction each circuit is process equivalent. All results are added together regardless of arrival order. Any circuits that are process equivalent destined for the same node can be mapped to the same stream net since there is no need to distinguish between them.

Communication patterns: A communication pattern is a group of circuits that are all mapped to the

same stream net and may co-exist in the network at the same time.

Send-receive disjoint: A section of processor code is send-receive disjoint if the sends and receives for outgoing and incoming circuits are not interleaved.

Guaranteed routing: A communication pattern is guaranteed routing if every circuit in the pattern can be routed before the first receive for that pattern is issued. For example, any pattern in which there is no contention for links (i.e. the routes are disjoint) is guaranteed routing.

Buffer conservation: A communication pattern is *strictly buffer conservative* if each processor in the pattern cannot stall on a send before it receives every data item destined for it in the pattern. Thus, a processor must issue every receive instruction for a pattern before it may issue a send that could overflow the buffer space for a circuit. In general, strict buffer conservation ensures that processors sink enough data to guarantee forward progress of all circuits in the communication pattern.

A communication pattern is *leniently buffer conservative* if each processor in the pattern cannot stall on a send unless the sum of the number of receives and the buffer space are greater than the number of sends it has issued. When a communication pattern is guaranteed routing, lenient buffer conservation is sufficient to ensure forward progress of the communication pattern because any processor stalls are only due to buffer constraints. Thus, if the overall buffer pool of the communication pattern does not overflow, at least one processor will be able to advance at any given time.

Correct, deadlock-free communication can be ensured by mapping circuits that are not process equivalent to different communication patterns. Communication patterns may overlap in time provided they are in different stream nets and processor communication activity in different patterns is strictly ordered. In general, processor code should be send-receive disjoint and strictly buffer conservative for a communication pattern. If the pattern is a guaranteed routing pattern, the code may be only leniently buffer conservative and is not required to be send-receive disjoint.

4 Simulation model

I performed simulation studies using a MIPS R2000 based multiprocessor simulator. The base machine was a 64-node (8x8), 2-dimensional mesh with MIPS R2000 processors at each node. Each processor had a 32kB, 4-word-per-line, direct-mapped, write-back, unified cache.

The network was fully simulated. Network links were full-duplex with 4-virtual channels per unidirectional physical link. The routing algorithm was deterministic e-cube routing[4, 9]. Thus, a total of four disjoint stream nets were available to the parallel programs. The network flow-control mechanism was wormhole routing using 16-bit flow-control digits (*flits*).

Table 1: *Benchmark results*

Benchmark	Cycles	Memory stalls	Network stalls
MM par	51,462	1,028.0	17.2
MM seq	2,862,227	133,190.0	0.0
FFT par	7,035	2,272.2	283.5
FFT seq	53,449	3,660.0	0.0
Gauss par	893,673	8,669.8	492,043.4
Gauss seq	14,704,485	1,966,150.0	0.0
LU par	459,554	4,130.9	8,396.6
LU seq	15,289,994	3,881,600.0	0.0

Table 2: *Software overhead*

Benchmark	Total bytes	Overhead instr	Instr/Byte
MM	1,032,448	515,359	0.500
FFT	12,288	10,445	0.850
Gauss	4,386,816	2,389,376	0.545
LU	2,747,136	1,385,799	0.504

5 Benchmarks

I implemented four algorithms using data streaming to demonstrate its effectiveness. The benchmarks used were matrix multiply (*MM*), fast Fourier transform (*FFT*), Gaussian elimination (*Gauss*), and LU decomposition.(*LU*). Each benchmark was optimized for data streaming. A more complete description of the benchmarks and their communication patterns is available in [7].

6 Results

Tables 1 and 2 tables summarize the results of the simulation studies on the benchmarks.

Overhead instructions were the total of all instructions used for network communication and for destination address calculation. We see that for each benchmark, the software overhead per byte of data transmitted was less than 0.9 instructions. This is a result of placing the network instructions directly into the program code and eliminating the buffering and context switch associated with library based communication methods.

The benchmarks display a variety of behaviors. First, *MM* is clearly the best benchmark for parallelization and for data streaming. The flow of data was easily arranged so that network latency was overlapped with useful instructions. As a result, on the average, PEs spent only 17.22 cycles idling on network instructions. The *FFT* algorithm is also very amenable to parallelization, albeit less so that *MM*. The problem distribution is not well balanced and, thus, PEs stall on network instructions an average of 283.5 cycles out of 7,035. *Gauss* proved to be the most difficult benchmark to parallelize. The algorithm is strongly sequential in nature and is inherently unbalanced. The available parallelism declines linearly during the forward elimination phase as rows are elim-

inated. As a result, the average network stall cycle count was over half of the total execution time. However, data streaming does reduce the software overhead associated with the algorithm. Interestingly, *LU* proved well suited to the techniques of data streaming. Despite the fact that *Gauss* and *LU* are similar in the types of operations performed, the well structured nature of *LU* greatly enhanced performance, the simplicity of the code and of the communication patterns. The number of network stall cycles for *LU* was 8,396.6 cycles per processor which was approximately twice the cache miss penalty.

It is interesting to note that the network stall cycles for *Gauss* and *LU* were not primarily due to communication latency. The stall cycles were primarily a result of load imbalance. The average circuit setup time for *Gauss* was 7.96 cycles and 8.56 cycles for *LU*.

7 Conclusions and future work

This paper has presented data streaming, a technique for very efficient communication in parallel architectures. Data streaming acheives very high performance communication by minimizing the software overhead required to transmit and receive data. Experimental results indicate that software overhead is less than one instruction per byte of data transmitted. This is significantly less than that achieved by other techniques[14, 11]. The cost of communication in terms of software overhead and in terms of idling due to network latency was lower for both *MM* and *FFT* than the memory stalls due to cache misses — indicating a very tight coupling between processors. The efficiency of data-streaming allowed a fast implementation of *Gauss* despite the strongly sequential nature of the algorithm. Another matrix operation benchmark *LU* proved very ammenable to the techniques of data streaming. The low software overhead of data streaming allows parallel programs to operate on finer-grained problems.

The architectural support given in this paper requires no microprocessor modification. However, modifying the microprocessor itself can also enhance the benefits of data streaming. The network interface could be implemented as a functional unit of a dynamic superscalar processor. In this case, the processor need not stall on blocked sends and unsatisfied receives. Instructions could be issued out-of-order allowing register reservations to preserve the data dependencies in the code. The implications of dynamic superscalar implementation of data-streaming need to be explored.

References

[1] S. Chandra, J. R. Larus, and A. Rogers. Where is time spent in message-passing and shared-memory programs? *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VI)*, October 1994.

[2] W. J. Dally. Performance of the Cray T3D communication network. *Personal communication*, February 1994.

[3] W. J. Dally, J. A. S. Fiske, J. S. Keen, R. A. Lethin, M. D. Noakes, P. R. Nuth, R. E. Davison, and G. A. Flyer. The message-driven processor. *IEEE Micro*, April 1992.

[4] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[5] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 1993.

[6] P. T. Gaughan. Design and analysis of fault-tolerant pipelined multicomputer networks. *PhD dissertation, Georgia Institute of Technology*, June 1994.

[7] P. T. Gaughan. Data streaming: A tightly coupled multicomputing paradigm. Technical Report UA/CARL-95/01, University of Alabama, February 1995.

[8] P. T. Gaughan, B. V. Dao, S. Yalamanchili, and D. E. Schimmel. Distributed deadlock-free routing in faulty pipelined k-ary n-cubes. Technical Report TR-GIT/CSRL-93/11, Computer Systems Research Laboratory, Georgia Tech, 1993.

[9] P. T. Gaughan and S. Yalamanchili. Adaptive routing protocols for hypercube interconnection networks. *IEEE Computer Magazine*, May 1993.

[10] J. Kuskin, D. Ofelt, M. Heinrich, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy. The Stanford FLASH Multiprocessor. *In Proceedings of the 21st International Symposium on Computer Architecture (ISCA '94)*, April 1994.

[11] R. S. Nikhil, G. M. Papadopoulos, and Arvind. *T: A Multithreaded Massively Parallel Architecture. *In Proceedings of the 19st International Symposium on Computer Architecture (ISCA '92)*, 1992.

[12] P. Nuth and W. J. Dally. The J-machine network. *In the Proceedings of the 1992 IEEE International Conference on Computer Design*, pages 420–423, 1992.

[13] Paragon XP/S Product Overview. *Intel Corporation*, 1991.

[14] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active messages: a mechanism for integrated communication and computation. *Proceedings of the 19th International Symposium on Computer Architecture (ISCA '92)*, February 1992.

1999 IEEE TENCON

SUPPORTING PREEMPTIVE MULTITHREADING IN THE ARX REAL-TIME OPERATING SYSTEM

Yangmin Seo*, Jungkeun Park*, Gwangil Jeon, and Seongsoo Hong***

*School of Electrical Engineering, **Dept. of Computer Engineering
Seoul National University, Seoul 151-742, Korea
E-mail: *{seoym,jkpark,sshong}@redwood.snu.ac.kr, **gijeon@ssrnet.snu.ac.kr .

Abstract

To support thread-based real-time computing, we propose a new multithreading architecture which includes virtual threads and scheduling event upcalls. We have implemented this architecture in the Arx real-time operating system. Our experimental results show that our user-level thread schemes outperform kernel threads found in important commercial operating systems without compromising the benefits of user-level threads.

I . INTRODUCTION

In modern operating systems, threads have emerged as an essential mechanism for structuring applications and representing concurrency [1]. They are extensively substituting traditional UNIX-like processes. More and more operating systems have been modified, and even created to support threaded programming. This trend is not an exception in real-time operating systems. POSIX specifies a priority-driven thread model in its real-time thread extension standards [6]. The reason behind such popularity is clear. Not only can threads offer the desired level of abstraction for representing concurrency, but also it can support efficient resource management and fast context switching. This helps real-time operating systems provide short and bounded dispatch latency for real-time threads. Moreover, threads allow programmers to structure programs such that multiple asynchronous events in programs can be effectively handled.

Threads can be implemented in various ways and incorporated into different places in a programming environment . They can be implemented as either a set of kernel functionalities or a set of library functions. In conventional programming, user-level threads are much more favored than kernel-level threads mainly for two reasons [1]: (1) kernel-level threads provide wrong abstraction for a programming environment, since they require that every application should be mapped onto a single thread implementation that the kernel provides; and (2) kernel-level threads incur more overhead than user-level threads, since with the kernel-level threads, processes must cross an extra user/kernel protection boundary for each thread operation.

On the other hand, in real-time programming, kernel-level threads are more often the choice of implementation than user-level threads, albeit those disadvantages of the kernel-level threads [7]. The kernel-level implementation can simplify thread scheduling and signal handling, which is a crucial property for real-time processing. With no kernel support, the user-level threads cannot deal with kernel-level thread blocking and signal propagation. Thus, when a user-level thread is blocked in kernel mode via a

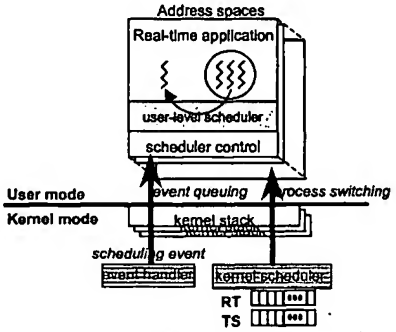

Figure 1 Overall architecture of the Arx kernel

multiprocessor issues and do not properly address real-time systems issues. For example, in [1], a costly scheduler activation is required for each event notification, since its user-level stack carries upcall arguments. Thus, if a new event occurs while an event is being processed in one scheduler activation, the kernel must create another and make an upcall interrupting the current scheduler activation. Though this scheme works well in multiprocessor systems, it incurs too much overhead for single processor embedded systems. Thus real-time operating systems have continued to support only kernel-level threads [7].

In this paper, we propose a new user-level multithreading scheme that is suitable for embedded real-time systems. It consists of *dynamic virtual thread binding* and *scheduling event upcall* mechanisms. A virtual thread is a kernel-level incarnation of a user-level thread that provides for a kernel-level execution environment. It is a passive entity which is temporarily bound to a user-level thread, when necessary. It is never scheduled by the kernel. The scheduling event upcall mechanism enables the kernel to notify user processes of kernel events such as thread blocking/unblocking and timer expiration. It thus allows real-time applications to preemptively schedule their own threads, as the kernel schedules kernel threads.

II. ARX MULTITHREADING SCHEME

The thread blocking anomaly of user-level threads occur since (1) scheduling events are not propagated to a user-level scheduler and (2) the blocking thread is put onto a sleep queue keeping the kernel stack designated to its process. We present two kernel mechanisms to solve these problems, respectively. They are scheduling event upcalls

0-7803-5739-6/99/$10.00©1999 IEEE.

and dynamic virtual thread binding. In designing these schemes, we put our greatest emphasis on reducing their overhead so that we can realize preemptive user-level threads for embedded real-time systems at an affordable cost.

A. Dynamic Virtual Thread Binding

A virtual thread plays a similar role to a kernel-level thread. It provides a user-level thread with a kernel stack and the kernel with an identifier to distinguish user-level threads running or being blocked in kernel mode. Unlike a kernel thread, however, it is a *dynamic* and *passive* entity. A virtual thread is just a kernel-level *incarnation* of a user-level thread. A user-level thread gets bound to a virtual thread when it enters into the kernel making a system call. Since a virtual thread is a passive entity, it does not multiplex user-level threads or it is not scheduled by the kernel.

It consists of a virtual thread identifier which is unique in a process, its state, and a kernel stack on which a user-level thread executes a system call. When a user-level thread gets blocked in the kernel, the associated virtual thread is enqueued onto the sleep channel. For each process, the kernel also maintains various fields in the process structure such as a current virtual thread, a pool of reserved virtual threads, and a list of active virtual threads bound to user-level threads. The kernel can identify user-level threads only through their bound virtual threads and does not affect thread scheduling at user level. Consequently, a user-level scheduler takes complete control over thread scheduling with the aid of scheduling event upcalls.

B. Scheduling Event Upcalls

For an application to control its threads at user level, it should be able to take appropriate actions on kernel events such as thread blocking/unblocking and timer expiration. To efficiently propagate such kernel events to an application, we propose a *scheduling event upcall* mechanism. In this scheme, a multithreaded process has a single vector entry to a user-level scheduler which processes scheduling events. Upon the occurrence of a scheduling event, the kernel upcalls a user-level scheduler to notify the event. Unlike normal function calls, upcall arguments are not pushed onto the stack of a user-level scheduler. Instead, they are passed through the *event queue* located in a kernel/user shared data structure we call a *scheduler control block (SCB)*. An SCB is mapped in the user's address space. Since events are queued by the kernel regardless of the execution state of the process, they can be delivered with a low latency even in a heavily loaded environment.

Event notification is carried out as follows. When an event occurs, the kernel first saves the context of the current thread. This is done differently according to the state of the current thread.

- When an event occurs while the current thread runs in kernel mode, the kernel takes out a virtual thread from the pool of reserved virtual threads (or creates a new one if it is empty), attaches it to the process, and makes it a current virtual thread. It then puts the old virtual thread identifier into the "interrupted thread context" field in SCB.
- Otherwise, the kernel copies the user's context onto the user stack, adjusts the old stack pointer in the register file, and saves it into the "interrupted thread context" field of SCB. (We assume that the thread package always saves a thread context onto its stack.)

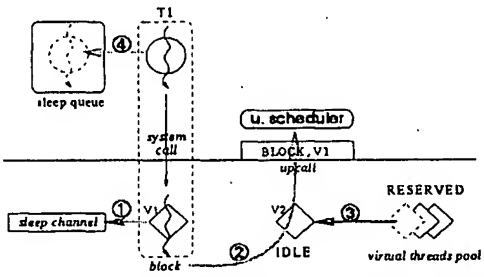

Figure 2 Scheduling event upcall example: when a thread blocks in a system call waiting for an I/O event.

After saving the context, the kernel makes an event, fills it with upcall arguments, and enqueues it into the event queue. If a user-level scheduler needs to be invoked, then the kernel builds its own return frame with the entry point of the user-level scheduler and its stack pointer. Finally, control is transferred to user mode through the return frame.

When a user-level scheduler is called by the kernel, it repeatedly takes out events from the event queue, until it empties the queue. For each event, it performs event-specific actions such as priority adjustment in priority scheduling. After consuming all events in the queue, it determines a thread to run next, and actually dispatches it.

As an illustrative example, Figure 2 shows the upcall procedure when a thread blocks in a system call waiting for an I/O event.

C. Event Types and Upcall Arguments

Our upcall scheme supports six types of scheduling events as follows.

- **BLOCK**: When a thread blocks in the kernel waiting for an event such as the completion of an I/O operation, the kernel immediately upcalls a BLOCK event along with the associated virtual thread identifier. For this type of event, the "interrupt thread context" field contains no meaningful information since no thread is interrupted by this upcall.
- **WAKEUP**: When a blocked thread wakes up in the kernel, a WAKEUP event is sent to a user-level scheduler along with the associated virtual thread identifier. The kernel also supplies the context information of the interrupted thread via SCB.
- **TIMER**: When a timer preset by a user-level scheduler expires, a TIMER event along with a timer delay value is sent to a user-level scheduler. When the kernel puts the event into the queue, it calculates

the difference between the requested timer value and the current time and writes it into the event.

- **SIGNAL**: When a signal is delivered to a process, the kernel upcalls a SIGNAL event along with signal information stored in the `siginfo` structure of the process structure.
- **EXIT**: When a process terminates via an exit()or exec() system call, the kernel sends this event to a user-level scheduler..
- **RESUME**: When a process is resumed after preemption, the kernel sends a user-level scheduler a RESUME upcall along with the duration of preemption. This event is used by the user-level scheduler to keep track of thread execution time.

D. Lock-Free Kernel-User Synchronization

Since the event queue is shared between a user-level scheduler and the kernel, there is a race between them. A naive solution to this synchronization problem is to use kernel supported locking primitives. However, this scheme is too expensive since it requires a user-level scheduler should make a system call every invocation. Instead, we use a lock-free scheme for synchronization between the kernel and a user process. In our scheme, we specially design a user-level scheduler in the following manner.

- A user-level scheduler consists of two subsections. In the upper section, it greedily consumes events in the event queue, until the queue gets empty. In the lower subsection, it selects a thread to run next.
- The execution of the lower subsection can be safely aborted. To make this possible, a user-level scheduler has its own private user stack but no thread control block.

In the SCB of a user process, there is the **lock** field which is set by a user-level thread when it executes non-reentrant code or wants to disable context switching. When an event occurs, the kernel can always enqueue it into the event queue regardless of the state of the user process. Before the kernel finishes an upcall, it checks if a user-scheduler should be invoked. It invokes a user-level scheduler, only if the **lock** value is zero and if the event queue was empty at the time of the upcall. The fact that the queue was not empty implies the user-level scheduler was interrupted by the kernel's upcall handler while dequeuing events from the queue. Thus, the kernel does not invoke a user-level scheduler in this case. If the queue was empty at the time of the upcall, then either a user-level thread was running, or a user-level scheduler was executing the lower section. In either case, the kernel invokes a new user-level scheduler. If there was a running user-level scheduler, the new invocation wipes out the context of the previous invocation since the new one uses the same stack as the old one. This is an entirely safe operation, since the user-level scheduler is designed to get aborted safely.

III. EXPERIMENTAL RESULTS

We have fully implemented the proposed schemes along with several other real-time system supports such as user-level interrupts, user-level timers, and real-time thread interfaces in the Arx real-time operating system that we have developed at Seoul National University. Arx is a stand-alone operating system possessing a fully preemptive real-time kernel, a POSIX compliant thread library, and multithread-safe standard I/O libraries. Arx is currently up and running on a Pentium PC and StrongARM NC (Net-work Computer). It supports the VFAT file system, the TCP/IP protocol suite, and the X11 R5 window system. Arx was designed to support flexible two-level scheduling, versatile multithreading, and efficient user-level I/O [3]. Specifically, scheduling in Arx is done in a nested fashion such that the kernel schedules processes and a user-level scheduler schedules threads in a process, as shown in Figure 1. The kernel scheduler runs the weighted fair queuing algorithm or its variants in that processes are scheduled and run according to their shares of CPU bandwidth [4].

We have conducted extensive experiments and measurements to show the performance of the proposed schemes implemented in the Arx real-time operating system. Our experimental machine was a PC with an Intel 100 MHz Pentium processor and 256K bytes of secondary cache. We made our measurements using a 64-bit cycle counter implemented in the Intel Pentium processor .

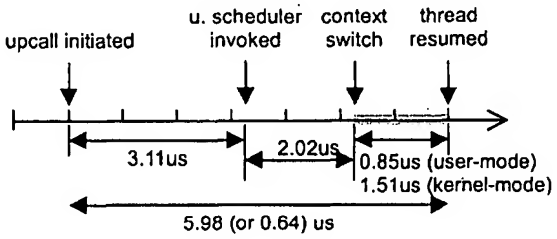

Figure 3 Upcall performance

We first measured an upcall latency. Since the Arx kernel relies heavily on upcalls, it is extremely important to minimize the upcall latency. Figure 3 shows the typical upcall performance in the Arx kernel with a typical user-level scheduler implementing fixed-priority scheduling with the round-robin policy. The upcall latency defined as time between when a kernel upcall handlers starts and when a user-level scheduler starts is 3.11us. In Figure 3, time to resume a thread differs only slightly depending on whether it is resumed in kernel mode or in user mode. This is because the resume() system call is specially designed such that it is handled by a dedicated trap handler which skips common signal check operations.

To show the performance benefits of our multithreading scheme over other operating systems, we have performed measurements with a multithreaded Java program in Arx, Solaris, and Windows 98. In these experiments, we used a 266 Pentium-II PC since we failed to install Solaris onto the 100 MHz Pentium PC. We also used a custom-made timer board with the timer resolution of 50ns since we could not access the Pentium event counters in user mode. In our benchmark Java program shown in Appendix of [5], each thread iterates 100,000 times writing its thread id into a global array protected by

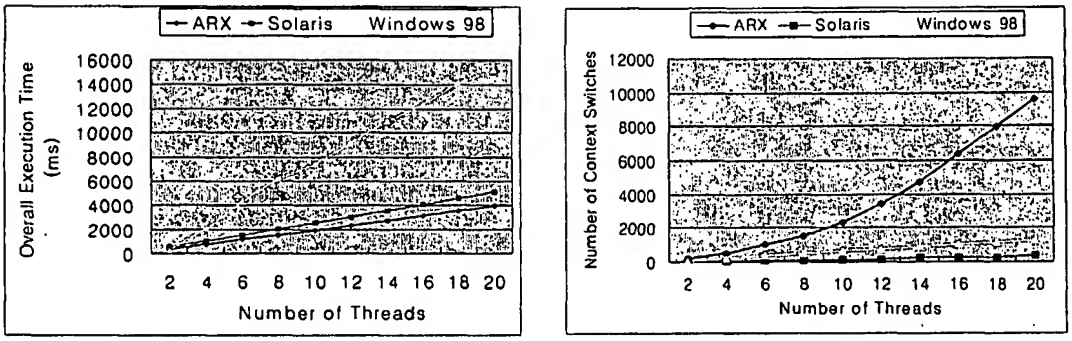

Figure 4 Performance of thread scheduling in Arx, Solaris, and Windows 98: (A) aggregate execution times of all threads, and (B) corresponding numbers of context switches.

a synchronized object. As a JVM, we ported and used Kaffe version 0.10.0 in Arx, Solaris and Windows 98 with the JIT mode being enabled.

We have measured the aggregate times of thread executions varying the number of threads. To observe only thread scheduling performance, we deliberately excluded time to load Java class files. In these experiments, all threads in the Java benchmark program had the same priority, and they were scheduled in a round robin fashion. Figure 4 summarizes the results of the experiments. Figure 4 (A) shows the aggregate execution times on three different operating systems, and Figure 4 (B) shows the numbers of context switches. As expected, Arx always incurred less scheduling overhead than Solaris and Windows 98 even though a lot more context switches happened in the benchmark program running on Arx.

Contrary to our expectations, the performance improvement of Arx over Solaris is marginal. This is because Solaris implements mutex lock/unlock primitives as user-level library functions even for kernel-level threads, and because the round robin quantum size is so large in Solaris that thread scheduling overhead contributes little to the aggregate execution times. Note that the quantum size was 1ms in Arx, while that was 200ms in Solaris. On the other hand, Arx outperformed Windows 98 by an order of magnitude. Unlike Solaris, Windows implements mutex primitives as system calls.

We speculate that Arx will outperform Solaris in a greater degree as the quantum size is reduced in Solaris. We can support this observation using a simple calculation as follows. When the number of threads was 20 in Solaris, the aggregate thread execution time was 5109ms with the quantum size 200ms. If the round robin quantum becomes 1ms, approximately 5,000 additional context switches will occur. Since the scheduling overhead measured in Solaris is 200us, 1000ms overhead will be added to the overall execution time. The increased execution time will recursively increase the number of context switches. Consequently, the overall execution time will be increased again. We can repeat this process until the overall execution time converges. The result is 6,374ms, which is 25% larger than the original execution time.

IV. CONCLUSION

These results show that user-level threads with our scheme yields much better performance than kernel-level threads while it can keep the desired properties of kernel-level threads.

REFERENCES

[1] T. E. Anderson and B. N. Bershad and E. D. Lazowska and H. M. Levy. Scheduler Activations: Effective Kernel Support for the User-level Management of Parallelism. Proceedings of ACM Symposium on Operating System Principles, pp. 95-109, 1991

[2] B. D. Marsh and M. L. Scott and T. J. LeBlanc and E. P. Markatos. First-Class User-Level Threads. Proceedings of ACM Symposium on Operating System Principles, pp. 110-121, 1991

[3] Y. Seo, J. Park, and S. Hong. Efficient User-Level I/O in the Arx Real-Time Operating System. ACM Workshop on Languages, Compilers, and Tools for Embedded Systems. pp. 162-171, Monteal, Canada, June 1998.

[4] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. IEEE/ACM Transactions on Networking, 1(3):344--357, December 1993.

[5] Y. Seo, J. Park, and S. Hong. Supporting Preemptive User-Level Threads for Embedded Real-Time Systems. SNU EE Technical Report, SNU-EE-TR-98-1, August 1998.

[6] Institute for Electrical and Electronic Engineers. POSIX P1003.4a, threads extension for portable operating systems.

[7] K.Schwan, H. Zhou, and A. Gheith. Multiprocessor real-time threads. Operating Systems Review, 26(1):54-65, January 1992.